REPORT ONR-CR215-228-2

FEATURE EXTRACTION AND RECOGNITION OF TWO-DIMENSIONAL DATA BY THE METHOD OF MOMENTS

R. C. GONZALEZ and J. M. HARRIS

ELECTRICAL ENGINEERING DEPARTMENT
UNIVERSITY OF TENNESSEE
KNOXVILLE, TN 37916

CONTRACT N00014-75-C-0545
ONR TASK 215-228

15 JANUARY 1977

104 p.

REPORT FOR THE PERIOD 1 OCT 75 - 1 OCT 76

Rept. for 1 Oct 75 - 1 Oct 76,

Approved for public release; distribution unlimited.

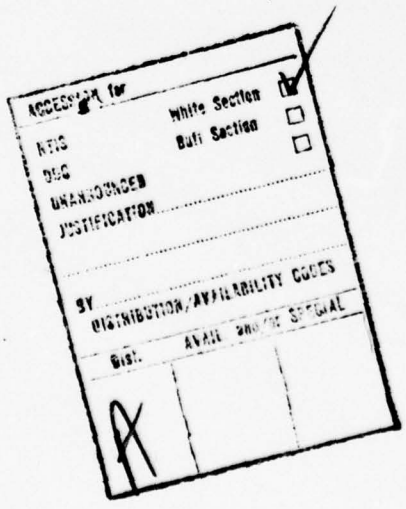
PREPARED FOR THE

OFFICE OF NAVAL RESEARCH • 800 N. QUINCY ST. • ARLINGTON • VA • 22217

ABSTRACT

A pattern recognition system applicable to two dimensional data is presented, and training algorithms for generating pattern classifiers are surveyed. The method of moments is used by the system as a feature extractor. The Mahalanobis distance measure is presented as a criterion for the selection of moment pairs to be used as descriptors. Experiments conducted using simulated high resolution radar images demonstrate the effectiveness of the system using unstructured data. Classification results for the system are compared to those of human interpreters.

TABLE OF CONTENTS



LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

Relatively recent developments in computer technology have made it possible to store or retreive data in less time than it takes light to travel 100 feet, and at a cost which makes these facilities available to almost anyone. Even with today's high speed computers, however, most businesses and data collection centers have a backlog of information to be coded and fed into these machines.

It is no longer adequate for computers to simply store and manipulate data in a mechanical fashion, but machines are now being required to make intelligent decisions about the data they process. Examples of some of the decisions being required of today's computers are character recognition, speech recognition, medical diagnosis, target recognition, and weather forecasting. Since the autonomous recognition of external stimuli promises to play a central role in any type of "intelligent" data processing task, the field of pattern recognition has been drawing much interest in recent years.

The problem of pattern recognition can be stated as the assignment of input data via certain features into a class with which the data shares common properties (Tou and Gonzalez, 1974).

There are two broad approaches to pattern recognition; decision-theoretic and syntactic. Though there are no well established rules governing which method produces optimal results when applied to a specific problem, experimentation has produced certain guidelines in

selecting which approach to use. If the data are well structured and spatial relationships are important in describing the pattern classes, then the syntactic approach promises optimal classification results. If, on the other hand, the data are best represented in a numerical form such as measurements or statistics, then the decision-theoretic approach is best applied. Outlines of both approaches are given in this chapter.

A. Approaches to Pattern Recognition System Design

The Syntactic Approach

The origin of formal language theory may be considered to be the development of mathematical models of grammars by Noam Chomsky. One of the original goals of linguists working in this area was to develop formal grammars capable of describing natural languages. From this work evolved the syntactic approach to pattern recognition. Syntactic pattern recognition has been applied to the specific problems of chromosome recognition, classification of cloud chamber patterns, and the recognition of geometric shapes. Since syntactic classification schemes derive their ability to discriminate between classes from the connectivity of patterns, it has also been the source of much interest in the study of scene analysis.

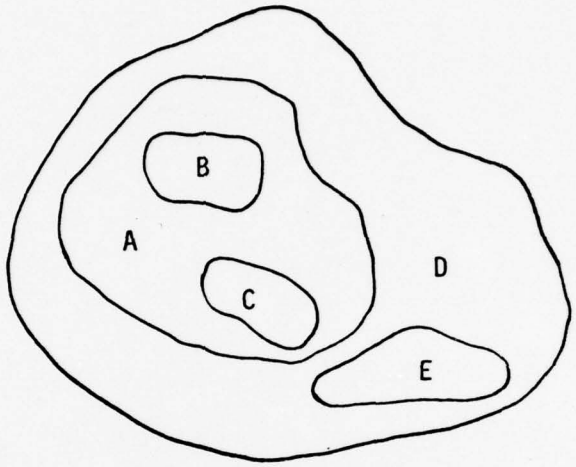
In the syntactic approach to pattern recognition, the patterns are specified via the use of primitives and productions. The primitives are the basic building blocks used in describing shape. For example, a set of primitives used in the description of two dimensional rectangular shapes may be the directed line segments $\uparrow$, $\rightarrow$, $\downarrow$, and $\leftarrow$. The productions

are the rules which may be applied to the primitives to produce a sentence (pattern) which belongs to a particular grammar (pattern class). A pattern may be represented as a string, tree, or graph.

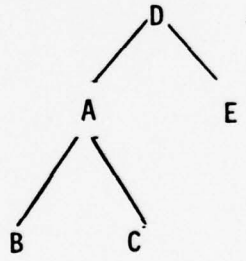
The training of a syntactic recognition system begins with the measurements representing the training data being used to build one of the structures mentioned above; for example, a tree. The training patterns are then used to construct a grammar for each class, which will properly generate the training patterns. If an unknown pattern is presented to the recognizer, the new pattern is subjected to the rules (productions) of each grammar, and is assigned to the class whose grammar produced the minimum number of errors while reproducing the given pattern.

An example of a tree representation of Figure 1-1(a) is shown in (b). The relationship used to derive the tree representation of this pattern is "inside of."

One of the earliest applications of the syntactic approach was the recognition of chromosomes. Ledley et al. (1965), developed two grammars used in dividing chromosomes into one of two classes according to their shapes. Figure 1-2 shows the primitives used, and strings which represented the chromosomes. The assumptions were made that the boundaries of the chromosomes formed closed figures and were traced in the clockwise direction. The strings which represented the chromosomes were then parsed against two grammars, one representing the submedian class of chromosomes, the other the telocentric class. Figure 1-2(b) shows two representative chromosomes and their string representations (Young and Calvery, 1974).

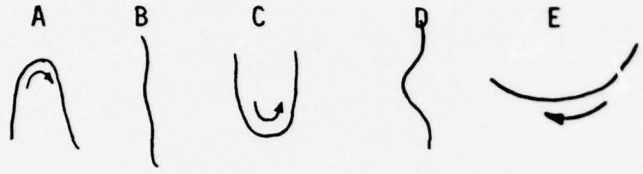


(a)

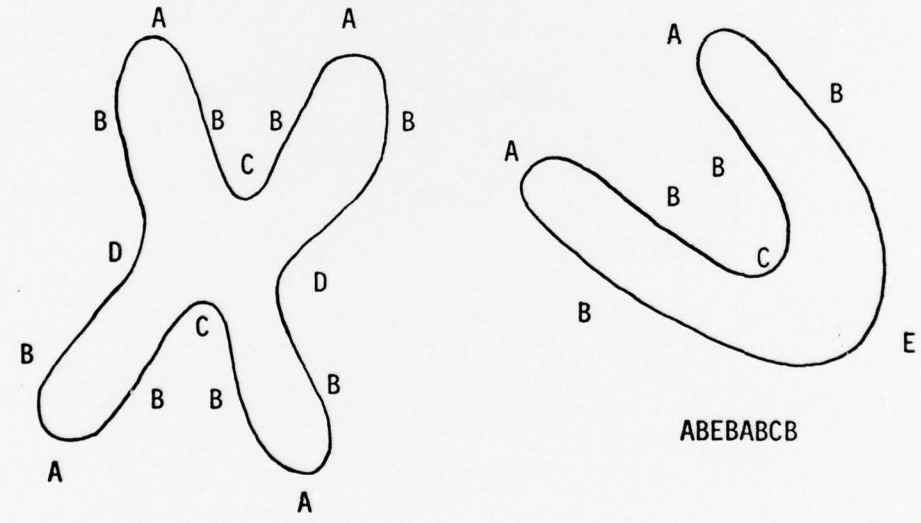


(b)

Figure 1-1. The tree representation of the pattern (a) is given in (b).



(a)



(b)

Figure 1-2. The primitives used in Ledley's chromosome recognition system are shown in (a), while chromosomes representative of the two classes and their string descriptions are illustrated in (b).

It should be noted that the syntactic approach to pattern recognition is viewed as a powerful technique primarily when the connectivity of patterns is important to the recognition of the objects. Whenever the patterns are not well structured, and connectivity relations do not contain significant amounts of discriminate information, then the following approach is best applied.

The Decision-Theoretic Approach

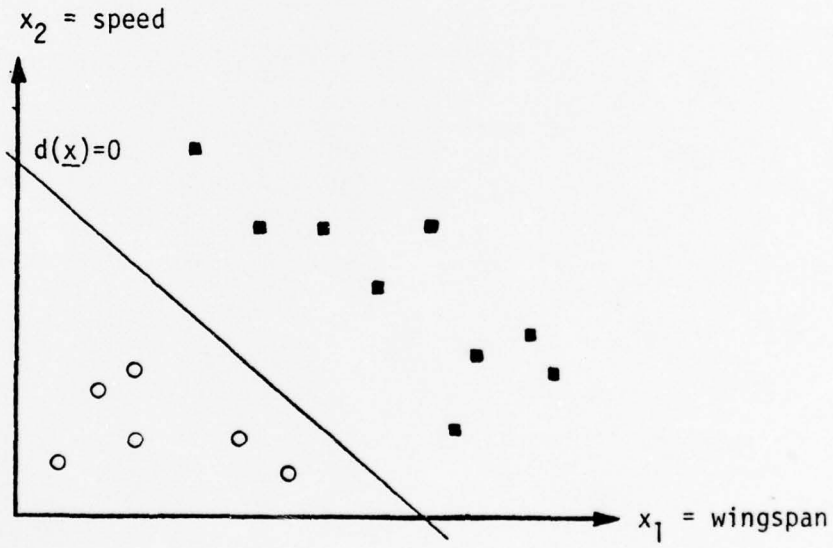
If patterns can be adequately represented by numerical information, then the decision-theoretic methods of pattern recognition are generally the best approach to the problem. In these methods, the numerical data used as features to describe a pattern are usually arranged in the form of a vector.

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \qquad (1\text{-}1)$$

where the elements x_i, i=1,2, . . ., n, are the measurements or features used in representing a given pattern.

The vector $\underline{x}$ may be viewed as a point in an n dimensional space. As a simple example, consider two classes, one whose members are the military airplanes of today, the other the airplanes of the 1920's. Figure 1-3(a) shows a hypothetical plot of the two classes using the wingspan as the only descriptive measure. As seen in the plot, the

(a)



(b)

Figure 1-3. The plot in (a) represents the aircraft using wingspan as the only descriptor, while (b) includes information concerning the maximum speed as well.

○ aircraft of the 1920's
■ aircraft of today

two classes overlap and may not be separated by a single line. If the dimension of the pattern space is increased by adding a second descriptor, say maximum speed, then the two classes may be plotted as in Figure 1-3(b). The line drawn between the two classes in the figure represents a suitable decision boundary for this example, since it properly separates the two classes. By increasing the dimension of the pattern space, it is always possible to produce accurate classification results if no two patterns of different classes are identical.

If the pattern classes are not linearly separable, such as in Figure 1-1(a), page 4, then the dimensions of the pattern vectors may be increased without increasing the complexity of the measurement device which produces $\underline{x}$. This is accomplished by forming the vector

$$\underline{x}^* = \begin{bmatrix} f_1(\underline{x}) \\ f_2(\underline{x}) \\ . \\ . \\ . \\ f_R(\underline{x}) \end{bmatrix} \tag{1-2}$$

where $f(\underline{x})$ is a real, single valued function of $\underline{x}$. The linear decision functions in the R dimensional space of Equation (1-2) are mapped into nonlinear decision functions in the n dimensional space of Equation (1-1), since $f(\underline{x})$ may be nonlinear in form.

Classification of patterns into their respective classes will be achieved in this investigation by evaluating M decision functions, and assigning the pattern to the class whose decision function is the maximum.

That is, we say a pattern vector $\underline{x}$ belongs to class ω_i if

$$d_i(\underline{x}) > d_j(\underline{x}), \tag{1-3}$$

for all j not equal to i. Normally, the decision functions are a weighted sum of the components of vectors such as

$$d_i(\underline{x}) = w_{i1}x_1 + w_{i2}x_2 + \ldots + w_{in}x_n + w_{in+1}. \tag{1-4}$$

This may be written in vector notation as

$$d_i(\underline{x}) = \underline{w}_i^T\underline{x}. \tag{1-5}$$

Note that to allow the vector notation to be used in Equation (1-5), the pattern $\underline{x}$ must be augmented so that the vector product is consistent. The augmented vector has the form

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \\ 1 \end{bmatrix} \tag{1-6}$$

and will be used without special mention whenever needed throughout the following chapters.

If there are M pattern classes, then it is desired to find M weight vectors which will minimize any classification errors. The weight vector $\underline{w}_i$ is an n+1 dimensional vector which represents class ω_i. In the learning phase of the recognition system, it is the goal of the training algorithms to find the $\underline{w}_i$, i=1, 2, . . . M, which will classify the training patterns with minimum error.

B. Problem Formulation

The main goal of this investigation was to develop a pattern recognition system with which to study the effectiveness of two-dimensional moment pairs as descriptors of radar images. Since the returns from a high resolution radar vary significantly with relatively small changes in aspect angle, the patterns are nonstructured in shape and do not lend themselves to the syntactic approach described above. The present study investigates the use of a decision-theoretic scheme which allows for interactive experimentation with modeled radar returns.

Figure 1-4 shows the general structure of the recognition system. Note that for experimentation purposes, the operator controls the training algorithms and classifiers. Once in field operation, the training algorithms would still require a human operator, but the classification system would run autonomously. The laboratory setup shown in Figure 1-4 provides considerable flexibility in the procedures for testing the hypotheses drawn during this investigation.

A brief outline of the topics discussed in subsequent chapters is as follows: Chapter 2 surveys the classification algorithms used in training the recognition system. The previous uses of moments in pattern recognition as they appear in the literature are discussed in Chapter 3, along with the application of the Mahalanobis distance as a criterion for feature selection. Chapter 4 describes the pattern recognition system as it was implemented, and gives the experimental results obtained with test data. The conclusions drawn from the experimental results, along with suggestions concerning future work are discussed in Chapter 5.

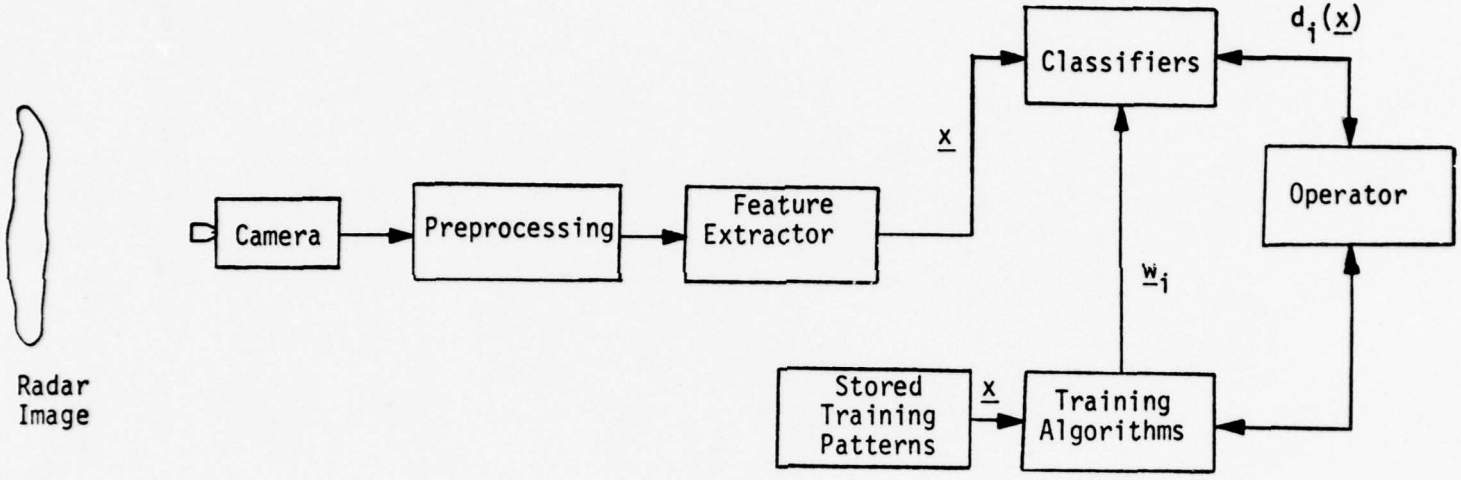


Figure 1-4. General diagram of pattern recognition system.

CHAPTER 2

PATTERN CLASSIFIERS

This chapter presents a discussion of the pattern classifiers used in this investigation. All of the classifiers mentioned are of the decision-theoretic type and may be divided into two basic categories--the statistical approach and the deterministic approach.

The Bayes classifier is a statistical classifier which derives decision functions based solely upon the statistics of the training patterns. If the statistics of the training patterns can be accurately specified, then the Bayes classifier yields a solution which minimizes the expected loss due to misclassification. This makes the Bayes classifier a valuable standard in the evaluation of results.

The perceptron and the least-mean-square-error (LMSE) algorithms both belong to the deterministic category of classification schemes. Each of these two classifiers is implemented by algorithms which learn a solution to the classification problem by iterating through the training patterns a finite number of times. The perceptron algorithm is easily implemented, but it is generally slow in reaching a solution during the training phase. A much more involved method of training is represented by the LMSE algorithm. In most separable cases, this algorithm will converge in a very few number of iterations, but its implementation is much more complicated and requires more memory in a computer than the perceptron classifier.

A comparison of these three classification approaches is presented in the following sections.

A. Bayes Classifier

The Bayes classifier used in this investigation is based on decision functions of the form

$$d_i(\underline{x}) = p(\underline{x}/\omega_i)\ p(\omega_i), \quad i=1,2, \ldots, M \tag{2-1}$$

where M is the number of classes. The terms $p(\omega_i)$ and $p(\underline{x}/\omega_i)$ are, respectively, the a priori probability and conditional probability density functions of the patterns of class ω_i. The decision function $d_i(\underline{x})$ which is the maximum corresponds to the minimum loss in classification (Tou and Gonzalez, 1974); therefore, the decision functions $d_1, d_2, \ldots, d_M$ are all computed and the unknown pattern $\underline{x}$ is assigned to class ω_i if $d_i(\underline{x})$ is the largest.

To apply Equation (2-1), it is necessary to determine the statistics of the pattern classes by specifying $p(\omega_i)$ and $p(\underline{x}/\omega_i)$. In a supervised learning environment, it is usually possible to use subjective judgment of physical properties in estimating the a priori probabilities. If for example, the classifier is to be used in determining the outcome of a toss of a coin, then $p(\omega_1)=p(\omega_2)=1/2$ since the results of the toss are equally probable. To estimate the probability density functions $p(\underline{x}/\omega_i)$, it is often necessary that a particular form of density such as the Gaussian or normal density be assumed. The multivariate normal density is

$$p(\underline{x}/\omega_i) = \frac{1}{(2\pi)^{n/2}|\underline{C}_i|^{1/2}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{m}_i)^T \underline{C}_i^{-1}(\underline{x}-\underline{m}_i)\right] \tag{2-2}$$

where $\underline{C}_i$ is the covariance matrix determined from the class population,

$\underline{m}_i$ is the mean of the class, and n is the dimension of the patterns. The term $\left|\underline{C}_i\right|$ is the determinant of the covariance matrix.

Due to the exponential form of the normal density function, it is more convenient to express Equation (2-1) as

$$d_i(\underline{x}) = \ln p(\omega_i) + \ln p(\underline{x}/\omega_i). \tag{2-3}$$

By substituting Equation (2-2) into Equation (2-3), the intermediate result

$$d_i(\underline{x}) = \ln p(\omega_i) - \frac{n}{2} \ln 2\pi - \frac{1}{2} \ln\left|\underline{C}_i\right| - \frac{1}{2}(\underline{x}-\underline{m}_i)^T \underline{C}_i^{-1}(\underline{x}-\underline{m}_i) \tag{2-4}$$

is obtained. Since the term $\frac{n}{2} \ln 2\pi$ is a common factor to all of the decision functions, it may be dropped from Equation (2-4), yielding the final form

$$d_i(\underline{x}) = \ln p(\omega_i) - \frac{1}{2} \ln\left|\underline{C}_i\right| - \frac{1}{2}(\underline{x}-\underline{m}_i)^T \underline{C}_i^{-1}(\underline{x}-\underline{m}_i). \tag{2-5}$$

The mean vectors ($\underline{m}_i$) and covariance matrices ($\underline{C}_i$) in Equation (2-5) are given by

$$\underline{m}_i = E_i\{\underline{x}\} \tag{2-6}$$

and

$$\underline{C}_i = E_i\{(\underline{x}-\underline{m}_i)(\underline{x}-\underline{m}_i)^T\} \tag{2-7}$$

where $E_i\{\cdot\}$ is the expected value in class ω_i. The mean vector may be estimated using the arithmetic average

$$\underline{m}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \underline{x}_{ij}$$

where $\underline{x}_{ij}$ is the jth pattern from class i and N_i is the total number of patterns in class ω_i. The covariance matrices can be similarly estimated by

$$\underline{C}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \underline{x}_{ij} \underline{x}_{ij}^T - \underline{m}_i \underline{m}_i^T. \qquad (2\text{-}9)$$

Equations (2-5), (2-6), and (2-7) completely describe the Bayes classifier for Gaussian data.

One of the advantages of the Bayes classifier is the speed with which it can be trained. To specify the decision functions as given in Equation (2-5) only one pass through the data is needed to estimate the covariance matrices and mean vectors. This information along with the a priori probabilities completes the training of the Bayes classifier. The decision boundary predicted by Equation (2-5) is a hyperquadric since there are no terms of higher than second order in $\underline{x}$. This limits the decision functions to a second degree system of equations, which may not be sufficient for separation of the classes. In order to evaluate the boundaries predicted by the Bayes classifier, the decision functions must be tested against the training set. If no classification errors occur, then the boundaries properly dichotomize the classes.

Example 2-1

As a numerical example of the Bayes classifier consider the two pattern classes ω_1: $\{(1,0,1)^T, (1,0,0)^T, (0,0,0)^T, (1,1,0)^T\}$ and ω_2: $\{(0,0,1)^T, (0,1,1)^T, (0,1,0)^T, (1,1,1)^T\}$. Applying Equation (2-8) directly yields

$$\underline{m}_1 = \frac{1}{4}\begin{bmatrix}3\\1\\1\end{bmatrix}, \text{ and } \underline{m}_2 = \frac{1}{4}\begin{bmatrix}1\\3\\3\end{bmatrix}.$$

Applying Equation (2-9),

$$\underline{C}_1 = \underline{C}_2 = \underline{C} = \frac{1}{16}\begin{bmatrix}3 & 1 & 1\\1 & 3 & -1\\1 & -1 & 3\end{bmatrix}$$

and

$$\underline{C}^{-1} = \begin{bmatrix} 8 & -4 & -4 \\ -4 & 8 & 4 \\ -4 & 4 & 8 \end{bmatrix}.$$

If we assume equal a priori probabilities, $p(\omega_1) = p(\omega_2) = 1/2$, then applying Equation (2-5) and dropping terms common to $d_1(\underline{x})$ and $d_2(\underline{x})$ yields

$$d_1(\underline{x}) = 4x_1 - 3/4$$

and

$$d_2(\underline{x}) = -4x_1 + 8x_2 + 8x_3 - \frac{11}{2}.$$

The decision boundary may be expressed as

$$d_1(\underline{x}) - d_2(\underline{x}) = 8x_1 - 8x_2 - 8x_3 + 4 = 0$$

and is plotted in Figure 2-1.

B. Deterministic Classifiers

The Perceptron Algorithm

During the early work in the field of artificial intelligence, Rosenblatt (1957) developed a set of machines known as perceptrons. These machines were developed in an effort to simulate human learning. From this work, the perceptron algorithm was developed which incorporates a mathematical approach to machine learning in a relatively easy to implement scheme.

The central goal of the perceptron algorithm is to learn a weight vector $\underline{w}$ such that

$$d(\underline{x}) = \underline{X}\,\underline{w} > 0, \qquad (2\text{-}10)$$

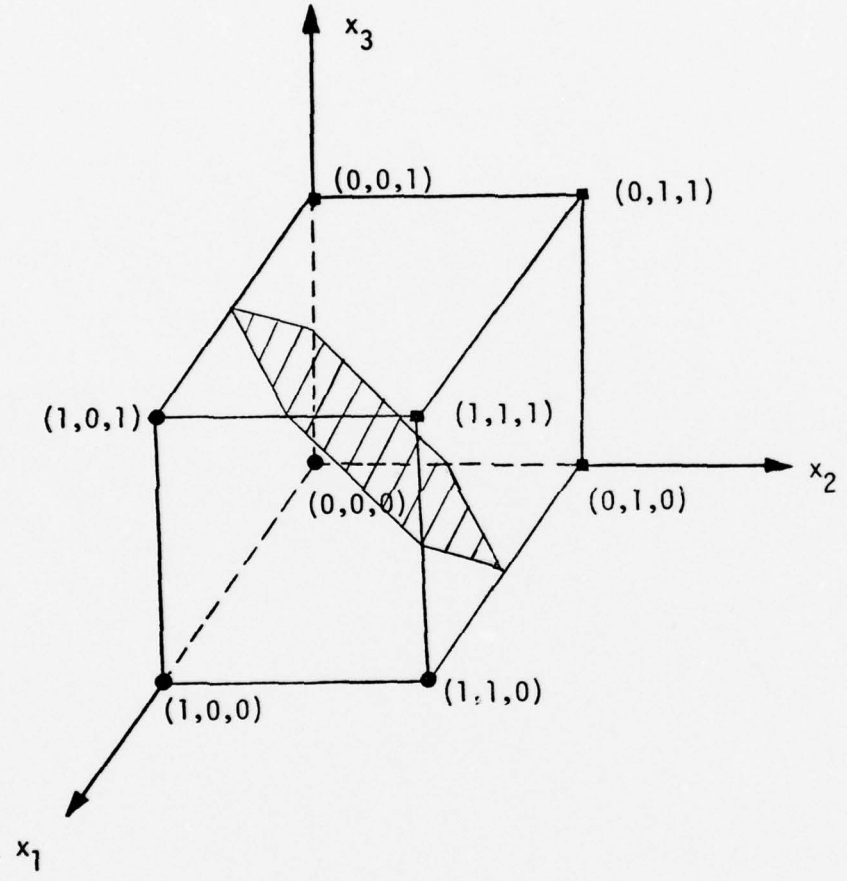


Figure 2-1. Plot of the patterns and decision boundary of Example 2-1.

for the two class case. In Equation (2-10), the term $\underline{X}$ may be expressed as

$$\underline{X} = \begin{bmatrix} \underline{x}_{11}^T \\ \underline{x}_{12}^T \\ \vdots \\ -\underline{x}_{21}^T \\ \underline{x}_{22}^T \\ \vdots \\ -\underline{x}_{2N}^T \end{bmatrix}, \tag{2-11}$$

where $\underline{x}_{ij}$ is the jth augmented pattern of class ω_i. Note that all of the patterns of class ω_2 have been multiplied by minus one. The perceptron algorithm solves the inequality of Equation (2-10) by using the reward-punishment concept. If a pattern is presented to the perceptron and classification is correct, the reward consists of not changing the weight vector $\underline{w}$. If, on the other hand, misclassification occurs, the term $c\underline{x}$ is added to the weight vector to form the new $\underline{w}$. For two pattern classes, this training scheme may be written as

$$\underline{w}(k+1) = \begin{cases} \underline{w}(k); \text{ if } \underline{X}\,\underline{w} > \underline{0} \\ \\ \underline{w}(k) = c\underline{x}_i(k),\ c > 0; \text{ if } \underline{w}^T\underline{x}_i \leq 0. \end{cases} \tag{2-12}$$

To expand the two class form of the algorithm to the multi-class case, Equation (2-12) must be generalized. By allowing each class to

be characterized by a single weight vector, the kth iterative step of the perceptron algorithm may be expressed as

$$\underline{w}_i(k+1) = \underline{w}_i(k) + c\underline{x}(k) \tag{2-13}$$

$$\underline{w}_\ell(k+1) = \underline{w}_\ell(k) - c\underline{x}(k) \tag{2-14}$$

$$\underline{w}_j(k+1) = \underline{w}_j(k) \qquad j \neq i,\ j \neq \ell \tag{2-15}$$

if $d_i(\underline{x}) \leq d_\ell(\underline{x})$, for $\underline{x} \in \omega_i$, occurred at step k, otherwise

$$\underline{w}_i(k+1) = \underline{w}_i(k),\ i=1,2,\ \ldots,\ M. \tag{2-16}$$

This algorithm converges whenever a complete iteration through the data produces no misclassifications. If the classes are separable, the perceptron algorithm will converge to a solution in a finite number of iterations, regardless of the choice of initial weight vectors.

As the patterns are examined by the perceptron algorithm, the weight vectors are adjusted to achieve correct classification. Unlike the Bayes classifier, the perceptron bases its decision functions on the patterns in the training set rather than the statistics of those patterns. This eliminates the assumptions required for the Bayes classifier and the problem of estimating the statistics of the training set. The perceptron is also free of the difficulties encountered whenever the inverse of a matrix must be calculated, as in the Bayes classifier, making implementation a much simpler task. Note also that since a solution is guaranteed if the classes are separable, the results of classification are always correct if convergence is reached and a test pattern is represented in the original training set.

The main disadvantage of the perceptron algorithm is that it

does not indicate if the pattern classes are separable, or how good a solution is obtained if the algorithm is stopped at an arbitrary iterative step. In the implementation of the perceptron algorithm for this study, a maximum iteration count is introduced to inhibit the algorithm from never terminating if no solution exists. The procedure also allows storage of the best set of weight vectors produced by the algorithm at step k. This set of weight vectors is produced by checking the accuracy of classification for each weight vector for all training patterns. If better classification results are obtained at this step than the results for the last weight vectors saved, then the new set of vectors replaces the old. By using this method, if the maximum learning sequence is reached without the algorithm converging, the best set of weight vectors are used as the solution. If the algorithm does not converge, or if a new pattern is added to the training set and retraining is required, then the algorithm starts with $\underline{w}_i(1)$ equal to the previous best results. If training has never been done on the pattern set then the initial weight vectors

$$\underline{w}_i(1) = \underline{0} \qquad i=1,2, \ldots, M \tag{2-17}$$

are chosen.

Example 2-2

As a numerical illustration of the perceptron algorithm, consider the three classes ω_1: $\{(0,0)^T\}$, ω_2: $\{(0,1)^T\}$, and ω_3: $\{(1,0)^T\}$. In order to implement the algorithm, the patterns must be augmented:

$$\underline{x}(1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad \underline{x}(2) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \text{ and } \underline{x}(3) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

Starting the algorithm with $\underline{w}_1(1) = \underline{w}_2(1) = \underline{w}_3(1) = \underline{0}$ and c=1, yields the following steps.

For k=1,

$$\underline{w}_1^T(1)\underline{x}(1) = 0$$

$$\underline{w}_2^T(1)\underline{x}(1) = 0$$

$$\underline{w}_3^T(1)\underline{x}(1) = 0.$$

Since all of the decision functions are equal, the following adjustments must be made to the weight vectors:

$$\underline{w}_1(2) = \underline{w}_1(1) + \underline{x}(1) = (0,0,1)^T$$

$$\underline{w}_2(2) = \underline{w}_2(1) - \underline{x}(1) = (0,0,-1)^T$$

$$\underline{w}_3(2) = \underline{w}_3(1) - \underline{x}(1) = (0,0,-1)^T.$$

For the next pattern $\underline{x}(2)$,

$$\underline{w}_1^T(2)\underline{x}(2) = 1$$

$$\underline{w}_2^T(2)\underline{x}(2) = -1$$

$$\underline{w}_3^T(2)\underline{x}(2) = -1.$$

Since all of the products are greater than or equal to $\underline{w}_2^T(2)\underline{x}(2)$, adjustments are needed to all of the weight vectors.

$$\underline{w}_1(3) = \underline{w}_1(2) - \underline{x}(2) = (0,-1,0)^T$$

$$\underline{w}_2(3) = \underline{w}_2(2) + \underline{x}(2) = (0,1,0)^T$$

$$\underline{w}_3(3) = \underline{w}_3(2) - \underline{x}(2) = (0,-1,-2)^T$$

Testing the weight vectors with $\underline{x}(3)$ yields

$$\underline{w}_1^T(3)\underline{x}(3) = 0$$

$$\underline{w}_2^T(3)\underline{x}(3) = 0$$

$$\underline{w}_3^T(3)\underline{x}(3) = -2.$$

Since the present weight vectors did not properly classify $\underline{x}(3)$, they are adjusted to:

$$\underline{w}_1(4) = \underline{w}_1(3) - \underline{x}(3) = (-1,-1,-1)^T$$

$$\underline{w}_2(4) = \underline{w}_2(3) - \underline{x}(3) = (-1,1,-1)^T$$

$$\underline{w}_3(4) = \underline{w}_3(3) + \underline{x}(3) = (1,0,-1)^T.$$

Since a complete, error free iteration through the data has not been obtained, the patterns must be recycled. Letting $\underline{x}(4) = \underline{x}(1)$, $\underline{x}(5) = \underline{x}(2)$, and $\underline{x}(6) = \underline{x}(3)$, then

$$\underline{w}_1^T(4)\underline{x}(4) = -1.$$

$$\underline{w}_2^T(4)\underline{x}(4) = -1$$

$$\underline{w}_3^T(4)\underline{x}(4) = -1.$$

Since all of the products are equal, the following adjustments are made:

$$\underline{w}_1(5) = \underline{w}_1(4) + \underline{x}(4) = (-1,-1,0)^T$$

$$\underline{w}_2(5) = \underline{w}_2(4) - \underline{x}(4) = (-1,1,-2)^T$$

$$\underline{w}_3(5) = \underline{w}_3(4) - \underline{x}(4) = (1,0,-2)^T.$$

For k=5, the products are

$$\underline{w}_1^T(5)\underline{x}(5) = -1$$

$$\underline{w}_2^T(5)\underline{x}(5) = -1$$

$$\underline{w}_3^T(5)\underline{x}(5) = -2.$$

Since $\underline{x}(5)\epsilon\omega_2$ was properly classified by $\underline{w}_3(5)$, no adjustment is made to $\underline{w}_3$.

$$\underline{w}_1(6) = \underline{w}_1(5) - \underline{x}(5) = (-1,-1,-1)^T$$

$$\underline{w}_2(6) = \underline{w}_2(5) + \underline{x}(5) = (-1,2,-1)^T$$

$$\underline{w}_3(6) = \underline{w}_3(5) = (1,0,-2)^T.$$

The algorithm proceeds in this manner making corrections to the weight vectors until iterative step k=11. The weight vectors have become

$$\underline{w}_1(11) = (-2,-1,0)^T$$

$$\underline{w}_2(11) = (-1,2,-2)^T$$

$$\underline{w}_3(11) = (2,0,-2)^T$$

and $\underline{x}(11) = \underline{x}(3)$, $x(12) = \underline{x}(1)$, and $\underline{x}(13) = \underline{x}(2)$

$$\underline{w}_1^T(11)\underline{x}(11) = -2$$

$$\underline{w}_2^T(11)\underline{x}(11) = -3$$

$$\underline{w}_3^T(11)\underline{x}(11) = 0.$$

Since $\underline{x}(11) \in \omega_3$, the pattern was classified correctly and

$$\underline{w}_1(12) = \underline{w}_1(11) = (-2,-1,0)^T$$

$$\underline{w}_2(12) = \underline{w}_2(11) = (-1,2,-2)^T$$

$$\underline{w}_3(12) = \underline{w}_3(11) = (2,0,-2)^T.$$

The next pattern, $\underline{x}(12) \in \omega_1$, is then tested.

$$\underline{w}_1^T(12)\underline{x}(12) = 0$$

$$\underline{w}_2^T(12)\underline{x}(12) = -2$$

$$\underline{w}_3^T(12)\underline{x}(12) = -2.$$

Again, the pattern was properly classified and

$$\underline{w}_1(13) = \underline{w}_1(12) = (-2,-1,0)^T$$

$$\underline{w}_2(13) = \underline{w}_2(12) = (-1,2,-2)^T$$

$$\underline{w}_3(13) = \underline{w}_3(12) = (2,0,-2)^T.$$

Trying to classify $\underline{x}(13) \in \omega_2$ yields

$$\underline{w}_1^T(13)\underline{x}(13) = -1$$

$$\underline{w}_2^T(13)\underline{x}(13) = 0$$

$$\underline{w}_3^T(13)\underline{x}(13) = -2.$$

Now, all the patterns in the training set have been properly classified, and the solution weigh vectors are:

$$\underline{w}_1 = \begin{bmatrix} -2 \\ -1 \\ 0 \end{bmatrix}, \quad \underline{w}_2 = \begin{bmatrix} -1 \\ 2 \\ -2 \end{bmatrix}, \text{ and } \underline{w}_3 = \begin{bmatrix} 2 \\ 0 \\ -2 \end{bmatrix}.$$

The resulting decision boundaries are shown in Figure 2-2.

The Least-Mean-Square-Error (LMSE) Algorithm

The algorithm described in this section not only develops a set of decision functions for classes which are separable, but it also indicates if no solution exists to a classification problem. The LMSE algorithm consists of the following set of iterative relations:

$$\underline{w}(1) = \underline{X}^{\#}\underline{b}(1), \quad b_i(1) > 0 \tag{2-18}$$

$$\underline{e}(k) = \underline{X}\,\underline{w}(k) - \underline{b}(k) \tag{2-19}$$

$$\underline{w}(k+1) = \underline{w}(k) + c\underline{X}^{\#}[\underline{e}(k) + |\underline{e}(k)|] \tag{2-20}$$

$$\underline{b}(k+1) = \underline{b}(k) + c[\underline{e}(k) + |\underline{e}(k)|], \tag{2-21}$$

where $|\underline{e}(k)|$ is the absolute value of each term of the error vector $\underline{e}(k)$. The $\underline{X}$ in Equation (2-19) is formed from the training patterns as given in Equation (2-11). The weight vector $\underline{w}$ is the solution to

$$\underline{X}\,\underline{w} \geq \underline{b} \tag{2-22}$$

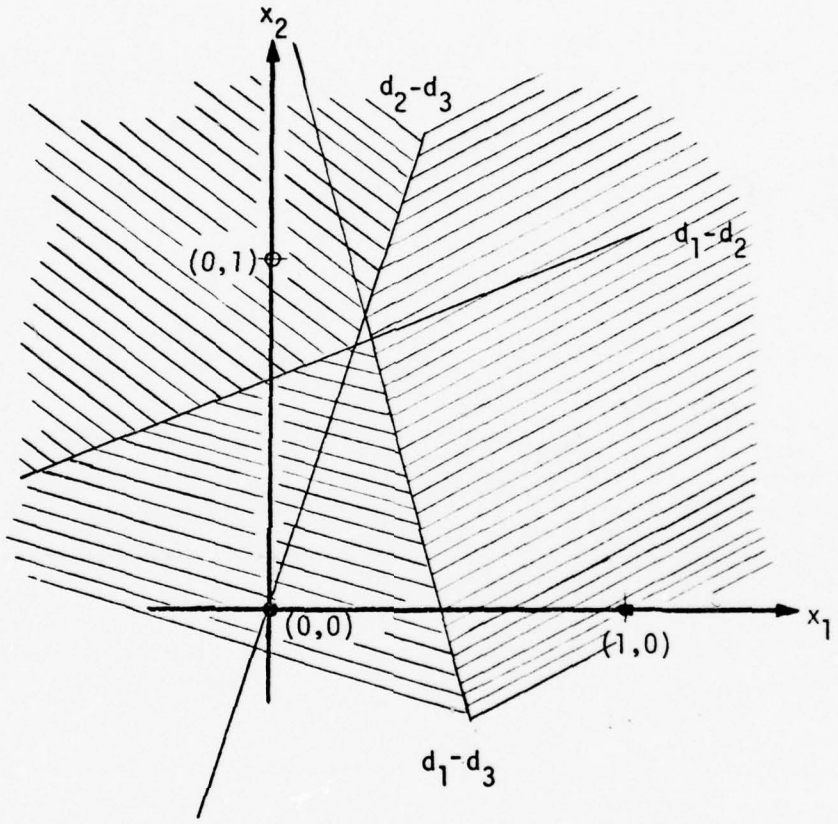


Figure 2-2. A plot of the decision boundaries from Example 2-2 and the regions described by the decision function.

which is equivalent to Equation (2-10) since $b_i(k) > 0$. The matrix $\underline{X}^{\#}$ is called the generalized inverse of $\underline{X}$ and may be expressed as

$$\underline{X}^{\#} = (\underline{X}^T\underline{X})^{-1}\ \underline{X}^T. \qquad (2\text{-}23)$$

The generalized inverse has the properties that it minimizes the sum of the squares of the residuals and that it minimizes the sum of squares of the unknowns (Noble, 1969). The derivation of this algorithm and its speed of convergence is based on these properties. The separability of the classes can be determined by examining the error vector $\underline{e}(k)$. If all the components of $\underline{e}(k)$ cease to be positive (but are not all zero) at any iterative step, then the classes are not separable by the specified decision boundary. The scalar constant c is required to be greater than zero and less than or equal to one for convergence. As with the perceptron algorithm, the LMSE is guaranteed to converge in a finite number of iterations if a solution exists (Tou and Gonzalez, 1974).

While it is a useful method for determining the existence of a solution, the LMSE algorithm does have some shortcomings. To generalize the LMSE algorithm for use with a multiclass problem, the classes must be considered pairwise. That is, a decision function must be found for each class which separates it from each other class. This involves applying the algorithm M(M-1)/2 times for an M class problem, which greatly increases the number of computations involved in training. The calculation of the generalized inverse also requires a large amount of memory in a digital computer. Consider the case of N training patterns of augmented dimension n. The matrix $\underline{X}$ then becomes an N by n matrix and the generalized inverse is an n by N matrix. These

dimensions soon surpass the memory size of a small computer for a large number of training samples. Another consideration is the existence of the inverse for the matrix $\underline{X}^T \underline{X}$. Since $\underline{X}^T \underline{X}$ is an n by n matrix, an inverse exists only if $\underline{X}^T \underline{X}$ is of rank n. If the training set consists of at least n well-distributed patterns, then the matrix can be shown to have an inverse.

Due to the above limitations of the LMSE algorithm, it was used only in determining the separability of pattern classes and not as a learning algorithm in this investigation. A numerical example of the LMSE algorithm follows:

Example 2-3

Consider the patterns for class ω_1: $\{(0,0)^T, (0,1)^T\}$ and ω_2: $\{(1,0)^T, (1,1)^T\}$. Augmenting the vectors and multiplying the patterns of ω_2 by -1 yields the matrix

$$\underline{X} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & -1 & -1 \end{bmatrix} .$$

The generalized inverse $\underline{X}^{\#} = (\underline{X}^T \underline{X})^{-1} \underline{X}^T$ is

$$\underline{X}^{\#} = \frac{1}{2} \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ 3/2 & 1/2 & -1/2 & 1/2 \end{bmatrix} .$$

Letting $\underline{b}(1) = (1,1,1,1)^T$ and c=1, and applying Equations (2-18, 19)

$$\underline{w}(1) = \underline{X}^{\#} \underline{b}(1) = (-2,0,1)^T$$

and

$$\underline{e}(1) = \underline{X} \underline{w}(1) - \underline{b}(1) = \underline{0}.$$

Since $\underline{X}\,\underline{w}(1) = (1,1,1,1)^T$, the algorithm has converged. Figure 2-3 shows the decision boundary found in this example.

Now consider the classes ω_1: $\{(0,0)^T, (1,1)^T\}$ and ω_2: $\{(0,1)^T, (1,0)^T\}$. Again letting c=1 and $\underline{b}(1) = \underline{1}$, we obtain

$$\underline{X} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & -1 & -1 \\ -1 & 0 & -1 \end{bmatrix}$$

and

$$\underline{X}^{\#} = (\underline{X}^T\underline{X})^{-1}\underline{X}^T = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 3/2 & -1/2 & -1/2 & -1/2 \end{bmatrix}.$$

The first weight vector is then

$$\underline{w}(1) = \underline{X}^{\#}\,\underline{b}(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

and the error vector

$$\underline{e}(1) = \underline{X}\,\underline{w}(1) - \underline{b}(1) = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}.$$

Since $\underline{e}_i(1)$ are all negative, the patterns are not linearly separable, and the algorithm terminates.

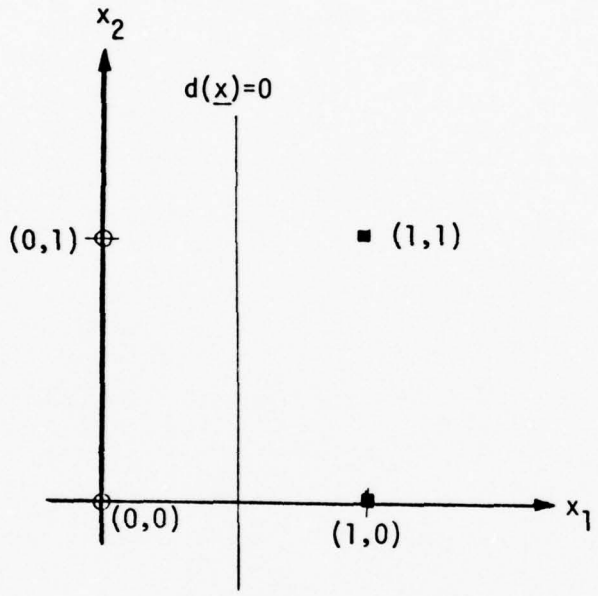


Figure 2-3. Illustration of the decision function found in Example 2-3.

CHAPTER 3

FEATURE EXTRACTION

The performance of a pattern recognition system is strongly dependent on the type of feature extractor used. Methods for feature extraction are often based on the intuition and the experience of the designer, gained through experimentation with a specific problem. The main guides to feature extraction are that the features should be insensitive to irrelevant variations, while emphasizing differences that are important in distinguishing between patterns of different classes (Duda, 1970).

Once a set of features have been chosen for use as descriptors, it is often desirable to reduce the dimension of the feature vectors. This can be accomplished by a transformation which maps the original feature space into one of lower dimensions while trying to optimize some criterion function (Tou and Gonzalez, 1974). It is also possible to simply delete any features which contain little or no information from the original feature set, thus forming a subset which produces equivalent classification results with less computation. It is often difficult to evaluate either the selection or dimensionality reduction of features since no single analytical criterion of performance exists.

Since the goal of this investigation is the automatic recognition of radar scatter returns, the selection of features must take into account several factors. The variability of the patterns and the high amount of noise encountered in this problem means that global

features would be the most useful in classification. One such set of features for two dimensional data can be obtained from the two dimensional moment pairs. The two dimensional central moment μ_{pq} of order p+q is given by

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q p(x,y)\, dxdy \qquad (3\text{-}1)$$

where $\bar{x}$ and $\bar{y}$ are the means of the population. It is assumed that p(x,y) is piecewise continuous and contains nonzero values only in the finite region of the xy plane. If the above assumption of finiteness holds, then a uniqueness theorem exists which states that the entire set of all μ_{pq} are defined by p(x,y) and, conversely, p(x,y) is uniquely defined by the set of all μ_{pq} (Hue, 1962). This uniqueness implies that if the patterns are not identical, enough moments may be chosen so that they form a discriminant set. Some noteworthy efforts in pattern recognition via the extraction of moments from the data are summarized below.

A. Summary of Previous Investigations Utilizing the Method of Moments

Character Recognition

One of the classical pattern recognition problems is the automatic recognition of alphanumeric characters. Some of the major problems encountered in designing such a system are variations in (1) size, (2) slant and rotation, (3) line thickness, (4) stroke regularity, (5) measurement noise, and (6) type fonts. There have

been two areas of interest generated by this problem concerning the application of moments. Both areas were stimulated by the way in which the invariant properties of moments may be applied.

Casey (1970) investigated a transformation which mapped a handprinted character into a pattern of more uniform appearance. In this study, the covariance matrix was used to derive a transformation matrix $\underline{A}$ such that

$$\underline{C}^* = \underline{A}\,\underline{C}\,\underline{A}^T \tag{3-2}$$

where

$$\underline{C}^* = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} . \tag{3-3}$$

The term σ^2 is the variance in both the x and y directions of the new pattern.

To obtain the normalized pattern $\underline{P}^*$, P is linearly transformed by

$$\underline{P}^* = \underline{A}\,\underline{P}. \tag{3-4}$$

The new pattern $\underline{P}^*$ has a covariance matrix $\underline{C}^*$ as described by Equation (3-3). Since all patterns are mapped into new patterns having identical diagonal moment matrices, the transformed patterns are identical except for reflections. It is interesting to note that the transformation is based upon moments of order two and manages to standardize the size and slant of the characters. This method does not effectively normalize line thickness and can amplify measurement noise. The results of this study extended through experimentation with the numerals 0-9 with a decrease in classification error of approximately 10% over non-standardized character sets. The motivation for such

a transformation was stimulated by the desire to apply template matching to handprinted as well as machine printed characters. Note that the moments were used to make the patterns of the same class similar, but were not used as the descriptors themselves.

In a study by Alt (1962) the moments of a character set of a particular type font were examined to evaluate their usefulness as features. The moments up to and including order six were calculated for 35 characters, each forming a pattern class. The higher order moments were normalized against the lower moments to make the patterns invariant to position and size. This reduced the number of useful moments to 22. The moments of order five were sufficient to distinguish between two characters as similar as O and Q of the type font used. Several classifiers were used in this study, including a finite automaton and a decision space approach.

Similar work has been done by Giuliano et al. (1961), in which moments were again used as the features for recognition. In this study the moments were normalized against (1) mass, (2) position, (3) orientation, (4) scale, and (5) perspective. The character set used for experimentation was again of a standard type font and a decision space classification was used. The results of the study showed that the first ten moments were sufficient for proper classification using a standard data set.

Ship Photo Interpretation

Smith and Wright (1971) investigated the use of moments as applied to ship photo interpretation. The images used were generated to resemble the high-contrast, low-resolution returns from a synthetic-

aperture radar. The study was limited to top views of images of three classes: merchant ships, destroyers, and submarines on the open sea with no returns from the water. The goal of the investigation was to estimate the length, width, and heading of the ships by using the method of moments.

Nonlinear functions were designed using a standard linear regression program by treating the powers of the moments as new variables in the linear combination. The order of the moments was kept below five since the higher order moments are more sensitive to random variations. Experiments were run with up to a 6-term cubic regression polynomial used for estimation, with the higher orders giving the best results. In order to evaluate the method, it was compared to the results of heuristic techniques and a human interpreter. The method of moments proved equivalent to or better than the heuristic techniques, and much easier to implement. When compared to the human interpreter, the method of moments was much more accurate in every category except for estimating the heading.

The four studies cited here represent the major uses of moments in automatic classification and interpretation schemes reported in the literature. In general, the use of moments is attractive because of their properties in minimizing the effects of size, location, and slant. The use of moments as descriptive features has also been attempted with modest success. The difficulty with using moments is that the selection of two dimensional moment pairs which best describe the data is not simple, and in practice has been confined to choosing an arbitrary number of low ordered moments and then

experimentally determining their usefulness. If any selectivity is applied, it is usually based only on physical reasoning and is of little value in the general case. In the following section, a criterion for measuring the value of features as descriptors and a procedure for estimating the criterion is presented.

B. A Criterion for Feature Selection

In order to effectively evaluate the performance of a set of features such as moments, it is necessary to establish a criterion with which to quantitatively measure the performance of the resulting feature set. If for example, a pattern classifier is designed to be used in deciding which one of two events have occurred, the features which best characterize the differences between the two events will normally produce optimal classification results. Since designing and implementing a pattern recognition system based on a set of features chosen without any a priori knowledge of their effectiveness will usually result in poor performance or, at best, redundant computation, it is desirable to have an indicator available which can predict the usefulness of a set of features as descriptors. A statistical distance measure is presented here as a tool for optimizing the choice of features.

Since the classifiers discussed in Chapter 2 utilize the concepts of a multidimensional space in order to generate decision boundaries which properly dichotomize the classes, it is meaningful to use a distance measure which includes information such as the Euclidean distances between the classes and the dispersion of the classes

about their means. One such distance measure is the Mahalanobis distance.

The Mahalanobis Distance

Given a multivariate normal distribution, the n dimensional Cartesian space on which the density is a constant forms an ellipsoid specified by the equation

$$D = (\underline{x} - \underline{m})^T \underline{C}^{-1} (\underline{x} - \underline{m}), \tag{3-5}$$

where $\underline{m}$ is the mean of the population and $\underline{C}$ is the covariance matrix, given in Equations (2-6) and (2-7), respectively. The ellipsoid described by a constant Mahalanobis distance has its center at $\underline{m}$ and its shape and orientation is specified by $\underline{C}$ (Cooley and Lohnes, 1971). Note that the Mahalanobis distance may be used to estimate the square of the distance from any point $\underline{x}$ to the center of the population.

The concept of the Mahalanobis distance and its usefulness as a guide to the selection of features may be better illustrated using uncorrelated data. Given a set of uncorrelated data, the covariances are all equal to zero and the covariance matrix $\underline{C}^*$ is given by

$$\underline{C}^* = \begin{bmatrix} \lambda_1^* & 0 & 0 \ldots 0 \\ 0 & \lambda_2^* & 0 \ldots 0 \\ 0 & 0 & \lambda_3^* \ldots 0 \\ \vdots & & \vdots \\ 0 & 0 & 0 \ldots \lambda_n^* \end{bmatrix} \tag{3-6}$$

where λ_j^* is the variance of component j in the feature space. The inverse of $\underline{C}^*$ is the diagonal matrix

$$\underline{C}^{*-1} = \text{diag}\left(\frac{1}{\lambda^*}\right). \tag{3-7}$$

Under these conditions, Equation (3-5) simplifies to

$$D^* = \sum_{j=1}^{n} \frac{1}{\lambda_j^*}(x_j^* - m_j^*)^2 \tag{3-8}$$

where x_j^* and m_j^* are the jth components of the n dimensional vectors $\underline{x}$ and $\underline{m}$ respectively.

In any pattern recognition problem, it is most desirable that the points which represent a pattern class be tightly clustered about their class mean and that the means be separated by the largest possible Euclidean distances in the feature space. These properties are inherent in the Mahalanobis distance since, to maximize D*, the variances along each component must be minimized and the square of the distance from the features and the mean of the population must be maximized. If two classes are considered, then Equation (3-8) may be written as

$$D^* = \sum_{j=1}^{n} \frac{1}{\lambda_j^*}(m_{1j}^* - m_{2j}^*)^2 \tag{3-9}$$

where m_{1j}^* and m_{2j}^* are, respectively, the jth components of the means of classes ω_1 and ω_2. This leads to the hypothesis that if the features can be chosen to maximize D*, then the degree of difficulty encountered in specifying a decision boundary in the feature space has been minimized. To generalize to the multiclass problem, Equation (3-9) may be written as

$$D_i^* = \sum_{j=1}^{n} \frac{1}{\lambda_j^*}(m_{ij}^* - \hat{m}_j^*)^2. \tag{3-10}$$

Here, m_{ij}^* is the jth component of the ith class and D_i^* is the Mahalanobis

distance of class ω_i. The term $\hat{m}_j^*$ is the jth component of the mean of the total population. Equation (3-10) may be interpreted for each descriptor as discussed in the next section.

Moment Pair Selection by the Mahalanobis Distance Criterion

If the feature or pattern vector $\underline{x}$ for each training measurement is given by

$$\underline{x} = \begin{bmatrix} \mu_{ij}(1) \\ \cdot \\ \cdot \\ \cdot \\ \mu_{k\ell}(n) \end{bmatrix} \tag{3-11}$$

where μ_{pq} is given in Equation (3-1) and n is the dimension of $\underline{x}$, then Equation (3-10) may be applied, yielding D_i^* as a measurement of the effectiveness of the n features taken as a whole. While the Mahalanobis distance is useful as a statistical classification measure, it would be more valuable if developed to be used in evaluating the moment pairs independently. It is indeed the case that Equation (3-10) suggests such a measure, say Δ_{ij}^*, where

$$\Delta_{ij}^* = \frac{1}{\lambda_j^*}(m_{ij}^* - \hat{m}_j^*)^2 . \tag{3-12}$$

The term Δ_{ij}^* characterizes the discriminating power of component j as it relates to class ω_i. If the moments which form $\underline{x}$ can be chosen which maximize Δ_{ij}^*, then D_i^* is maximized, and the moments which best characterize the differences between the classes have been determined.

While the development of the Mahalanobis distance for uncorrelated data is very useful, it is not always feasible or desirable to

decorrelate the features. The Mahalanobis distance in its general form as stated by Equation (3-5) would prove to be more useful if a distance measure Δ_{ij} can be established which will provide similar information as that obtained from Δ^*_{ij}.

Since a goal of this investigation is to independently measure the worth of each feature as a descriptor, Equation (3-5) may be rewritten to include the effect of increasing the dimensionality of the feature vectors. By allowing b to denote the dimension of the vectors and matrices, Equation (3-5) becomes

$$D(b) = [\underline{m}_1(b) - \underline{m}_2(b)]^T \underline{C}^{-1}(b)[\underline{m}_1(b) - \underline{m}_2(b)], \qquad (3\text{-}13)$$

$$b = 1,2, \ldots n$$

for the two class problem. It can be shown (Appendices A and B) that the Mahalanobis distances in the decoupled and original spaces are equal. This equality leads to

$$D(b) = D^*(b)$$

$$= \sum_{j=1}^{b} \Delta^*_j, \qquad (3\text{-}14)$$

where

$$\Delta^*_j = \frac{1}{\lambda^*_j}(m^*_{1j} - m^*_{2j})^2. \qquad (3\text{-}15)$$

By defining Δ_b in terms of D(b) as

$$\Delta_b = D(b) - D(b\text{-}1), \qquad (3\text{-}16)$$

a measure in the original space may now be related to Δ_b^* in the decoupled space by

$$\Delta_b = D(b) - D(b-1)$$

$$= \sum_{k=1}^{b} \Delta_k^* - \sum_{\ell=1}^{b-1} \Delta_\ell^*$$

$$\Delta_b = \Delta_b^*. \tag{3-17}$$

This shows the Mahalanobis distance to be a cumulative measure formed by the sum of the Δ_b's of Equation (3-16). If the features are chosen to maximize the Δ_b's, then the Mahalanobis distance will also be maximized.

Equation (3-13) and (3-16) may be generalized to the multiclass case by forming each equation so that it corresponds to a separate class. The multiclass case forms are

$$D_i(b) = [\underline{m}_i(b) - \hat{\underline{m}}(b)]^T \underline{C}^{-1}(b) [\underline{m}_i(b) - \hat{\underline{m}}(b)] \tag{3-18}$$

and

$$\Delta_{ib} = D_i(b) - D_i(b-1), \tag{3-19}$$

where $\underline{m}_i(b)$ and $\hat{\underline{m}}(b)$ are b dimensional vectors representing the mean of class ω_i and the mean of the population, respectively. The covariance matrix $\underline{C}(b)$ is a matrix of order b formed over all classes.

The formulation of the Mahalanobis distance by Equations (3-18) and (3-19) provides a method for the evaluation and selection of features which have not been decorrelated. Figure 3-1 shows a plot of the Mahalanobis distance as a function of increasing dimension. In this

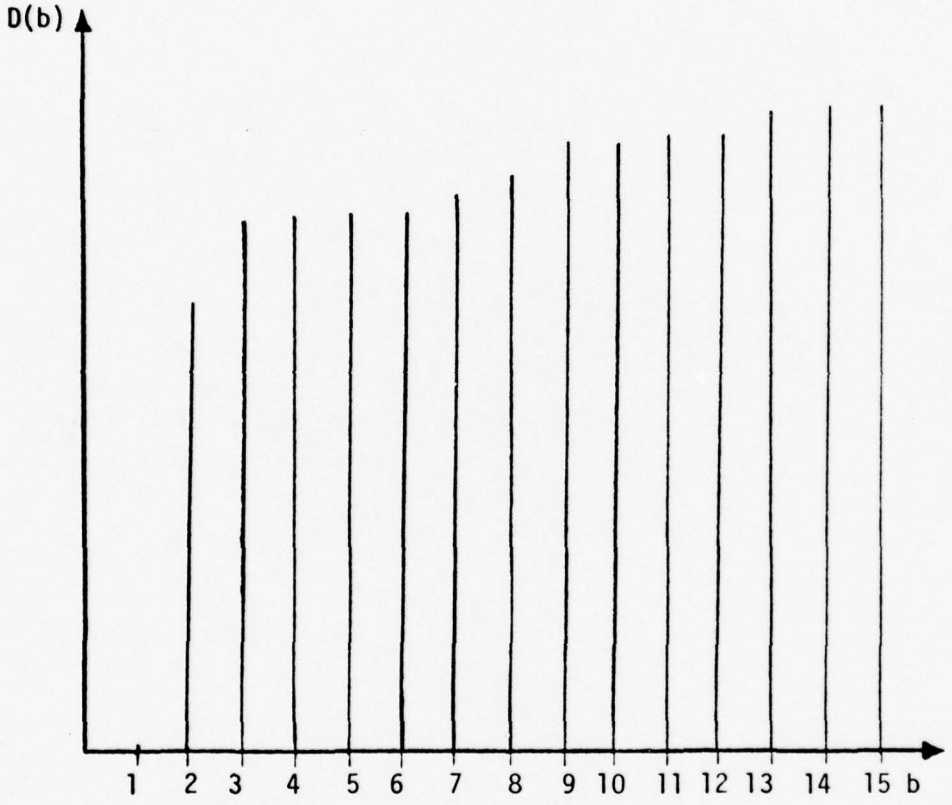


Figure 3-1. Graph of Mahalanobis distance between the means of two classes.

particular example, the maximum values of Δ_{ib} occur for b=2,3,9, and 13. (Classification tests were run using all 15 original moments and only the 4 moments which corresponded to b=2,3,9, and 13. The two classes were found separable in both cases, but only 27% of the calculations were needed for the 4 moments compared with the original 15.)

The selection of moments based on the Mahalanobis distance is not as simple when the multi-class case is considered since a moment pair which proves very useful in discriminating between two classes, may not be of any value in separating the other classes. When this situation occurs, trade-offs must be made between the complexity of calculations involved and the efficiency of the recognition system.

Computational Considerations

It is of interest to examine the complexity of the calculations involved in applying the foregoing Mahalanobis distance criterion. To evaluate D(b), b=1,2, . . .,n, directly requires the inversion of n matrices of order 1,2, . . .,n. The actual number of calculations required to obtain the inverse of a nth order matrix will vary depending on the pivoting strategy used, but Noble (1969) estimates that the number of multiplications involved is of the order n^3 and the number of additions is approximately n^3-2n^2+n. Using these estimates as guidelines, we see that calculating $D_i(b)$, for b=1,2, . . .,100, and i=1,2, . . .,10 requires a total of 255,025,000 floating point multiplications and 248,308,500 additions. The direct calculation of the Mahalanobis distance by computer will also usually require a large amount of memory. For the above example, it would take 40,000 bytes of memory to hold the inverse covariance matrix alone, and another 4,400

bytes for the mean vectors. All of the above figures are based on a standard single precision Fortran implementation. While it is not unreasonable to expect the large computers to be able to handle arrays and computations of this size, time on such machines is expensive. Often a smaller machine must be used, utilizing a peripheral bulk storage device to hold intermediate results. While this is usually a less expensive arrangement, the time for data transfers involved with such a system normally makes the execution times unbearably long.

Another common problem in the direct implementation of the Mahalanobis distance is the ill conditioning of the covariance matrix which usually occurs if the number of patterns is small. It can be shown (Anderson, 1958) that if K patterns of dimension n are chosen from a normal distribution, then the probability that the inverse of the nth order covariance matrix exists is one if $K \geq n$. Often in practice, K must be in the order of ten times n to produce a non-singular covariance matrix. This means that for our example, the maximum dimension is 100 and therefore, 1000 patterns should be available for the calculations of the covariance matrices.

It follows from the foregoing discussion, that an approximation to the Mahalanobis distance would be a desirable tool. Such a procedure is discussed below.

An Approximation to the Mahalanobis Distance

The Mahalanobis distance for uncorrelated data has already been shown to be much simpler than that for correlated data. This simplicity arises from the fact that the covariance matrix is diagonal, thus

making calculation of the inverse a simple task. In this investigation, the first order approximation given by

$$\hat{D}_i(N) = \sum_{j=1}^{N} \frac{1}{\lambda_j}(m_{ij} - \hat{m}_j)^2 \qquad (3\text{-}20)$$

has proven useful for evaluating feature vectors of dimension larger than N=30. (Note that Equation (3-20) is equivalent to Equation (3-10) if the data are uncorrelated.) Figure 3-2 shows a plot of $\hat{D}(N)$ for the same classes as the plot in Figure 3-1. Note that even though the values of the estimated distances are much different than those of Figure 3-1, most of the prominent changes in $\hat{D}(N)$(Δ_{ij}'s) are clearly visible in Figure 3-2. Of course, the accuracy of the estimate depends directly on the amount of correlation between the components. Since the covariance matrix contains information concerning the correlation of the data, the selection of features which exhibit a large correlation should be avoided to minimize the amount of error in the estimation of the Mahalanobis distance. If it is felt that these features are of importance in properly discriminating between classes, then it may be necessary to test them using the more complex but exact representation of the Mahalanobis distance.

Conclusions

The implementation of the Mahalanobis distance for use as a tool in the selection of features has been discussed in general in this chapter. A physical interpretation has been given the distance measure by examining D_i^* in the decoupled space. This lead to an approximation of the Mahalanobis distance assuming decorrelated data.

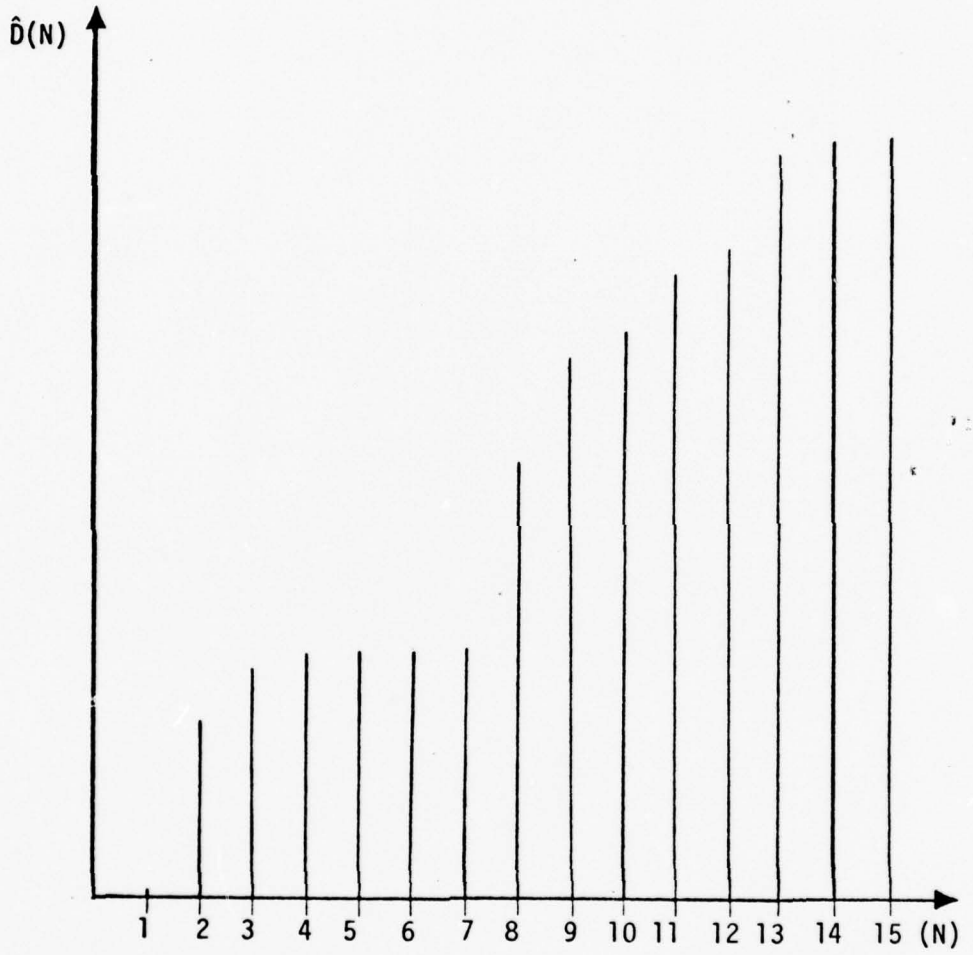


Figure 3-2. A graph of the estimated Mahalanobis distance between two class means.

The next chapter applies the method of moments and the criterion function discussed here to the specific problem of recognizing high resolution radar images.

CHAPTER 4

SYSTEM IMPLEMENTATION AND EXPERIMENTAL RESULTS

The pattern recognition system was implemented on a Digital Equipment Corporation PDP 11/40 minicomputer. The processor has 56K bytes of core memory and is supported by a real time operating system, which utilizes two 2.5 megabyte disk packs. The peripherals used by the system include a nine track magtape, an image digitizer, lineprinter, video terminals, and a video display generator. All of the peripherals mentioned would not be needed for a field implementation of the system, but were used in this investigation because of their interactive capabilities during training.

A. System Implementation

During the development of the pattern recognition system, some of the main objectives were to allow for interactions with the operator, and to allow maximum flexibility in dealing with varied applications. Due to the implementation of these features, the system offers a modular construction which may be altered to suit the needs of a specific problem. The computer programs which comprise the system may be divided into three categories. The programs which are used to generate the sample radar images will be discussed in this chapter first. The second program preprocesses the sample images and computes their moments, which are used by the third program. This last program performs the training required to generate decision functions, and then allows classification of patterns.

There are several other programs which are used as tools in developing the system, and will be discussed only as they relate to specific experiments.

The program which creates the sample radar images uses an image digitizer, which is equipped with a joystick with two degrees of freedom for inputing coordinates from a 512 by 512 image raster pattern. The operator inputs the drawing scale of the image to be stored and the size of the expected backscatter from each reflection point to the program. Then, by placing the joystick over each reflection point, the coordinates of the point are read by the computer. A Gaussian data cluster, centered about the reflection point, is then superimposed on the image when it is stored. The variances for the clusters are calculated by the program so that the desired reflection size is obtained at each point. Each image generated may have a maximum of 2,000 points.

Once the image has been created, the operator has the option of viewing the image via the video display terminal, or creating a data file on disk which is suitable for display via an available display program. If the operator is satisfied with the image, it may be stored on disk in a contiguous file, and another image sampled. If the image is saved, it is stored on a disk file along with an identification number, date of creation, reflection size, drawing scale, and up to 48 other images.

The second program extracts the moment pairs from a given image. The operator inputs an identifying class number, the name of the class, and the indices of the moment pairs desired. Images are then extracted from the specified disk files and are rotated into one of two standard positions, as discussed in Appendix A. Each image is then scaled by a

constant which was experimentally determined. This scaling is necessary to keep the numbers involved within the range of the computer, and does not affect the classification results. The images are extracted from the disk files by the operator specifying the identification number stored with each image. Of course, the images may be read from any of the available image files, allowing for concatenation of feature vectors from any image file into a single data structure forming a class.

The two dimensional moment pairs calculated by this program are given by

$$\mu_{pq} = \frac{1}{L} \sum_{i=1}^{L} (x_i - m_x)^p \, (y_i - m_y)^q, \qquad (4\text{-}1)$$

where L is the total number of data points in the image. The terms x_i and y_i are the ith coordinate pair, while m_x and m_y are the mean coordinates of the images, and p and q are the indices of the p+q order moment. A maximum of 31 moment pairs may be calculated for each image, with the maximum moment index value being 127. The identification number assigned the image is appended to the descriptors creating a 32 dimensional vector which is then stored on disk.

The vectors described above are stored in groups of 255 or less, along with a label record describing the file. It is assumed that all of the vectors stored in one of these files represent one pattern class. The label record, therefore, contains an identifying class number, the name of the class, the indices used in calculating the moments, the date created, and the number of vectors in the file. These files are in a form suitable for use as training data by the next program.

The main program of the recognition system provides the capabilities of training on stored data, and classifying data which has been processed by the programs just discussed. This program is also capable of obtaining new patterns from the image digitizer and preparing them for classification, or adding them to the data which has previously been stored. This is by far the most involved program in the pattern recognition system requiring a total of 23 subroutines, and 8,674 bytes of common array storage. The program contains five overlay segments which share the same memory. The system is interactive, allowing specification of files, algorithms used and the execution time allowed for the completion of a task.

A maximum of ten classes may be handled by the system at one time. The name of the training file for each class is stored at the beginning of the program, and each file is checked for compatibility with the other files prior to any other execution. If the dimensions of the vectors do not match, or if different moment indices are used in the training files, the error is flagged and the program halts. The names of the files containing any decision functions computed during past runs, or used to store the results of the present experiment, are entered from the console and stored for future use without operator intervention.

From this point, the operator has several options. If training files were entered in the initial dialogue, then the system asks the operator if training should take place. If no training files were entered, then the program continues to the classification section. Assuming that training is desired, the option of one or both of two algorithms is available.

If training is desired using the perceptron algorithm, the system asks for input of the incremental constant c, the maximum number of iterations allowed, and the maximum allowable training time. The system then sets up a file on disk, using the name entered earlier, for storage of the resulting weight vectors. Training then proceeds according to the algorithm given in Chapter 2. If the algorithm converges, a message is output to the terminal giving the number of iterations required for convergence, and the solution vectors are stored on disk. If convergence is not achieved within the given time limit, or the maximum number of iterations allowed, then the iteration count along with the number of patterns properly classified by the best solution thus far is output to the operator. The same information along with the name of the class is stored with the partial solution on disk. This procedure allows the operator to return to the system later, and restart the training sequence from the same step at which it stopped.

If training is required using the Bayes classifier discussed in Chapter 2, then the system calculates the mean vector and the covariance matrix of each class. The user then inputs the a priori probabilities for each class at the request of the system. This information, along with the name and number of the class, is stored on disk. Often if the training set is small, the covariance matrix is singular. Whenever this occurs, a message is sent to the operator, and control returns to the main program.

It should be noted that the LMSE algorithm has also been implemented for use with the system. This algorithm was not made an integral part of the main program due to the large array space needed for its implementation.

After the training sequence is either completed or bypassed, the system asks the operator if classification of a pattern is desired. If the operator replies affirmatively, then the pattern is either input from the digitizer or an existing disk file. In either case, the option of displaying the image via the video display terminal is available.

Once the image is made available to the processor, it is transformed according to the procedure described in Appendix A, and scaled. The moments are then calculated according to the indices of the training patterns, and the pattern vector is formed. Assuming a set of weight vectors exists for both algorithms, the operator may specify which classification scheme to use.

The system outputs the classification results to the operator by specifying the class number chosen and the name of the class. The operator must then indicate to the system if any classification errors occured. If an error did occur, the pattern vector may be added to the training set and either one or both of the algorithms may be retrained, if desired. Once retraining has completed, classification with the pattern producing the error may be tried again.

As presently implemented, the system provides a flexible, interactive method for testing the hypotheses developed by the user. The flexibility of being able to group several patterns into large classes, and to divide the classes into subclasses, allows the operator to test numerous classification arrangements for a given problem. The interaction of the computer hardward and the flow of data through the system as it relates to the operator is illustrated in Figure 4-1. A sample of the interactive capabilities of the system is presented in the following examples. All operator responses are underlined.

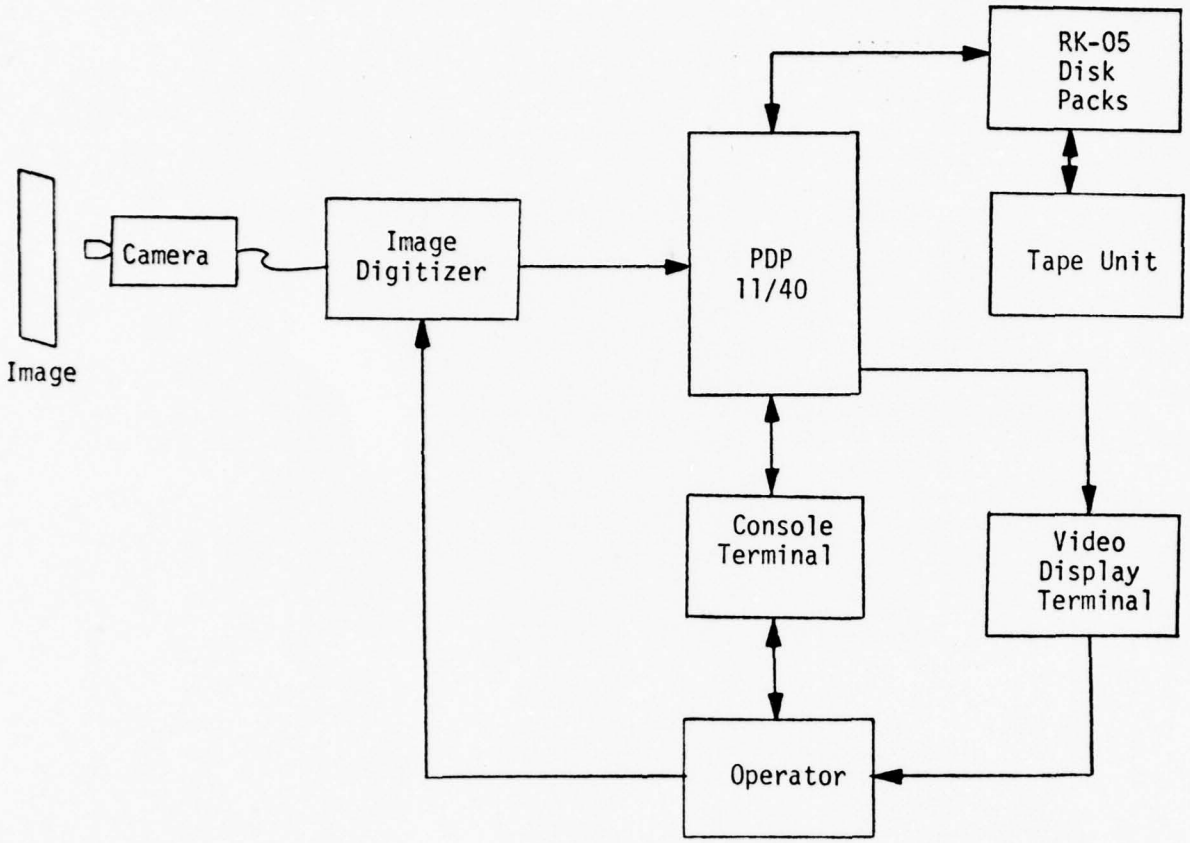


Figure 4-1. The general flow of information during operation of the system program.

Example 4-1

Once the initial dialogue has allowed the operator to specify the names of the files needed during the present session, the training section is entered.

```
DO YOU WANT TO TRAIN? Y

PERCEPTRON? Y

ENTER POSITIVE CORRECTION CONSTANT: 1.

ENTER MAXIMUM NUMBER OF ITERATIONS: 10

ENTER MAXIMUM TRAINING TIME ... HR,MIN,SEC FORMAT.
0,15,0

THE PERCEPTRON CONVERGED IN 8 ITERATIONS.

BAYES?
```

If the perceptron algorithm had not converged, the following message would have been printed.

```
AFTER 10 ITERATIONS, THE BEST SOLUTION
FOUND WAS 19 OF 20 PATTERNS CLASSIFIED
CORRECTLY.

BAYES?
```

If training is also required for the Bayes classifier, the following dialogue takes place.

```
BAYES? Y

ENTER PROBABILITY OF CLASS 1.
NAME OF CLASS IS B-737.
.5

ENTER PROBABILITY OF CLASS 2.
NAME OF CLASS IS F-111.
.5
```

The preceeding dialogue would require decision functions to be stored on the disk for both the Bayes and perceptron classifiers.

Example 4-2

After the decision functions have been computed and stored for future use, the system allows the operator to classify new patterns, as shown below.

DO YOU WANT TO CLASSIFY A NEW PATTERN? Y

IS PATTERN STORED ON DISK? Y

If the operator had answered N to this question, the program would assume that the pattern would come from the image digitizer. The dialogue required for the retrieval of a pattern from disk continues:

ENTER NAME OF DESIRED FILE.
PATTERN.TST

ENTER I.D. OF DESIRED PATTERN: 103

DISPLAY PATTERN? Y

Pattern 103 would be found stored on disk in a file named PATTERN.TST and would be displayed on the video display terminal. The pattern is then rotated, and the moments are extracted. The method of classification is then specified as follows.

DO YOU WANT A BAYES CLASSIFICATION? Y

DO YOU WANT A PERCEPTRON CLASSIFICATION? Y

If we assume that the pattern belonged to class 1, the following dialogue might occur.

THE PERCEPTRON ASSIGNED THE PATTERN TO CLASS 1

THE PATTERN IS A B-737.

CORRECT? Y

THE BAYES ASSIGNED THE PATTERN TO CLASS 2

THE PATTERN IS A F-111.

CORRECT? N

DO YOU WANT TO RETRAIN THE BAYES? Y

Since the Bayes classifier assigned the pattern to the wrong class and the operator requested training to be reinitiated for the Bayes, the pattern would be stored on disk with the other training files. The same dialogue for the Bayes classifier in Example 4-1 would then be issued by the program during the retraining sequence.

These two examples illustrate the versatility of the system, and the way it interacts with the user. The results discussed in the remainder of this chapter were obtained using this and related programs.

B. Experimental Results

Radar Data Model

The data used to test the recognition system were simulated to resemble the returns from a high resolution, synthetic aperture radar. Since real radar images were not available, the images were created by estimating the location of the specular reflections from airplanes, along with the reflection due to creeping waves. (Peebles, 1976). The resolution of the radar was (conservatively) set at ten feet in both the range and cross-range. Once the reflection points were located a two-dimensional Gaussian data cluster with a standard deviation of approximately five feet was superimposed on each reflection point. If the reflection was determined to be elongated, the variance of the cluster along the axis of elongation was increased so that the Gaussian cluster represented the true return more closely. A sample of the resulting image is shown in Figure 4-2.

Figure 4-2. Sample of modeled data.

Since modeling the data by this method is a slow, tedious task, only eight airplanes were considered. Approximately 14 images of each plane were created. These images represented changes in aspect angle of 45° in two geometric planes. This produced a total of 103 sample images, which were used in training and classification tests.

Table 4-1 lists the airplanes used, along with their wingspans. There are two main categories: fighters, and transports, with the latter category including cargo planes and bombers. Since human interpreters usually base their decisions on the size of the returns, the planes used in the experiments were chosen to allow for worst-case conditions in testing the system, with some of the wingspans differing only by one to four feet.

The data were generated under the assumption that there was little or no noise other than that produced by the reflection points on the object of interest. In practice, this can be achieved by time-integrating the images for human interpreters. It is also assumed that there is only one aircraft in the image at a time. More than one aircraft in the field of view of the radar could be handled by applying a clustering technique to extract the aircraft from a more complex image. Rotation and translation of the images are taken into consideration by the procedures described in Appendix A.

A brief outline of the experiments and topics discussed in the remainder of the chapter is as follows. The first thirty moments of the aircraft presented in Table 4-1 were calculated and their Mahalanobis distances were plotted. These distances were then used to determine which of the thirty moments to use in dealing with these

TABLE 4-1. Airplanes Used in the Experiments and Their Dimensions

Name and Manufacturer	Type*	Nationality	Wingspan
Mikoyan MIG-21	Fighter	Russian	23.46'
McDonnell-Douglass A-4M	Fighter	USA	27.5'
Mikoyan MIG-25	Fighter	Russian	40'
General Dynamics F111	Fighter	USA	63'
Tupelov TU-22	Transport	Russian	90.875'
Boeing 737	Transport	USA	93'
Antonov An-22	Transport	Russian	211.25'
Lockheed C5A	Transport	USA	222.71'

*Transport includes bombers and cargo airplanes.

particular patterns. The aircraft were divided into several groups, with training and classification experiments being conducted on the various subdivisions of the classes. An experiment is discussed which compares the classification results of the system to those of human interpreters. Finally, the Mahalanobis distance criterion is considered again, showing the error involved when the distance is estimated by assuming decoupled data.

Applying the Mahalanobis Distance

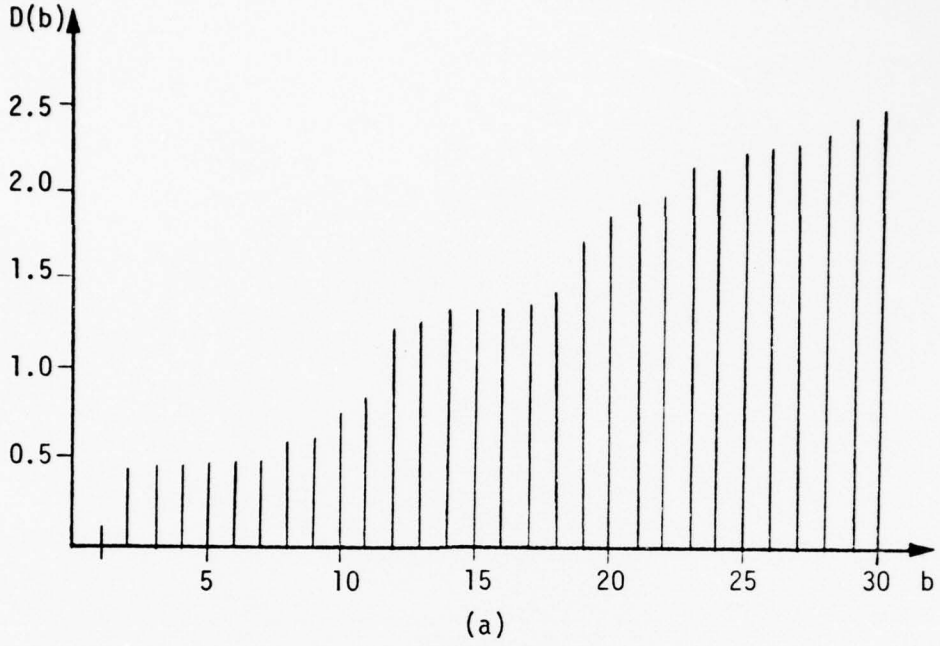
To determine which moment pairs to use, the Mahalanobis distance was plotted for the first thirty moments of the aircraft listed in Table 4-1. The indices of the moment pairs are listed for reference in Table 4-2. From these thirty moments, a subset was chosen by applying the Mahalanobis distance criterion.

Figures 4-3(a) - (f) show plots of the Mahalanobis distance for each of the eight types of aircraft, treating each type as a separate class. The following procedure was used to determine which the thirty moment pairs would be used for Experiments No.'s 1 through 4 in this chapter.

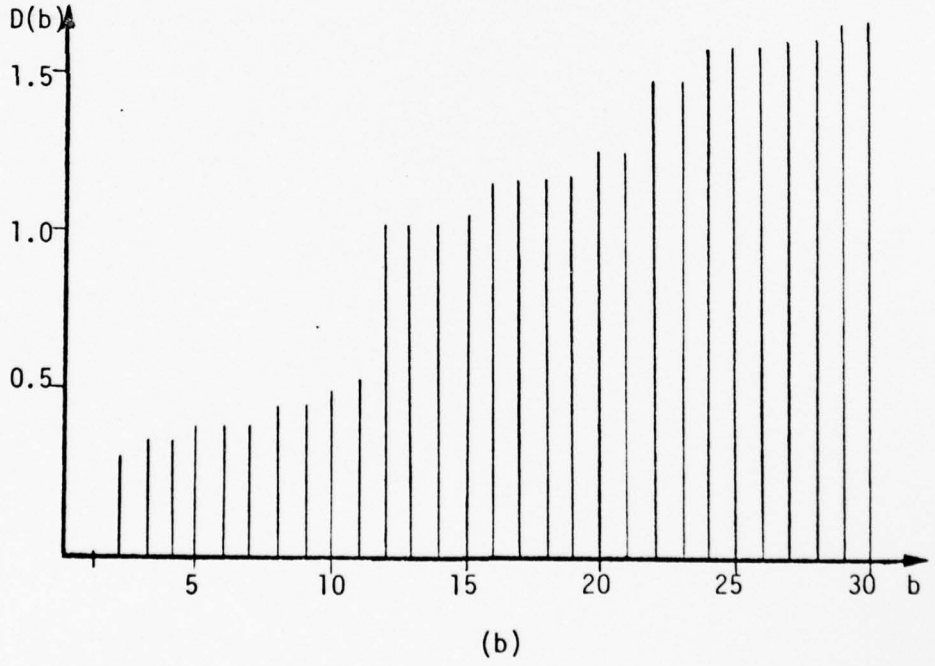
In Chapter 3, it was determined that to maximize the Mahalanobis distance between class means, the descriptors which produced the maximum gradients (Δ_{ib}'s) should be selected. In order to apply this criterion in a meaningful way, only the Δ_{ib}'s which represented at least 5% of the total Mahalanobis distance were considered. This reduced the number of moment pairs which were considered meaningful from thirty to twenty. The number of moment pairs used in the experiments was further reduced by keeping only those of the twenty

TABLE 4-2. Indices of the Two-Dimensional Moment Pairs Calculated for the Mahalanobis Distance Plots

Order	x Index	y Index
2	1	1
2	0	2
2	2	0
3	1	2
3	2	1
3	3	0
3	0	3
4	2	2
4	1	3
4	3	1
4	4	0
4	0	4
5	1	4
5	4	1
5	2	3
5	3	2
5	0	5
5	5	0
6	1	5
6	5	1
6	2	4
6	4	2
6	3	3
6	6	0
6	0	6
7	1	6
7	6	1
7	5	2
7	2	5
7	3	4

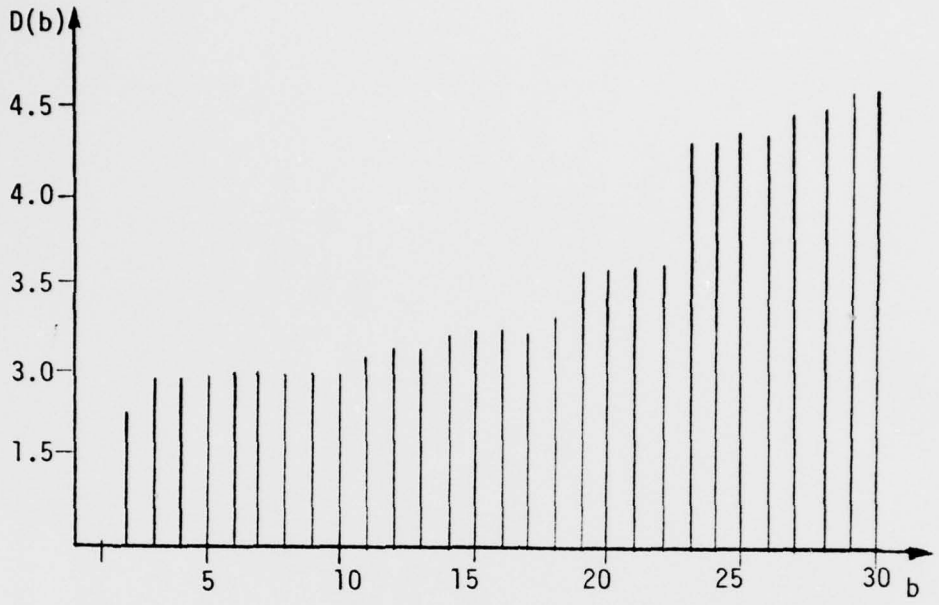


(a)

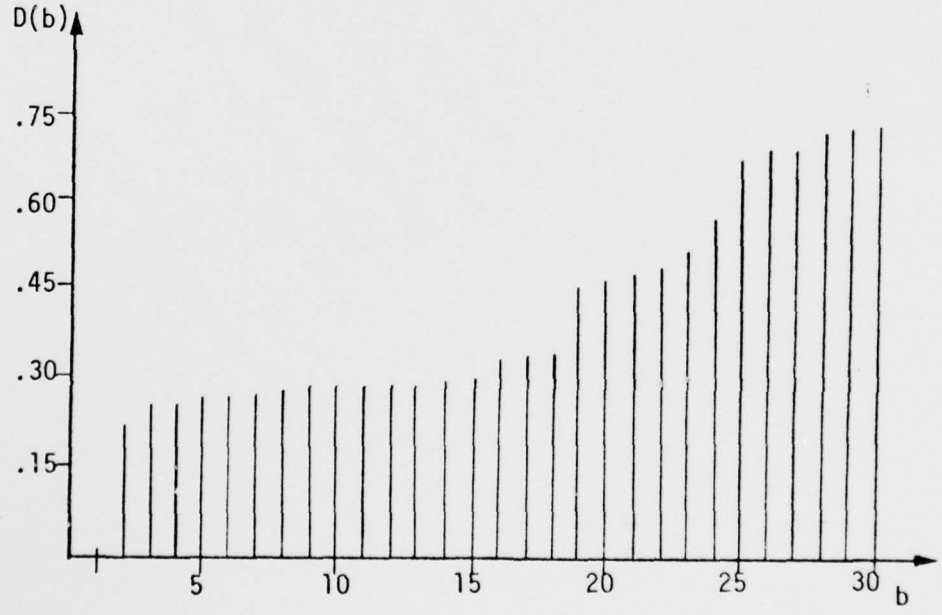


(b)

Figure 4-3. The Mahalanobis distance for the aircraft listed in Table 4-1, page 59, are plotted here. The aircraft are (a) MIG-21, (b) A-4M, (c) MIG-25, (d) FB-111A, (e) TU-22, (f) B-737, (g) AN-22, and (h) C5A.

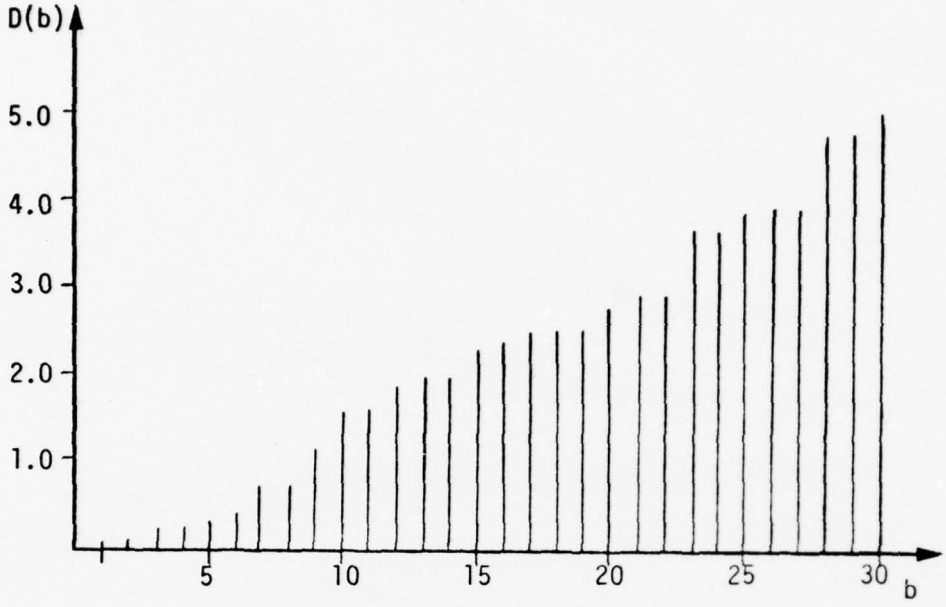


(c)

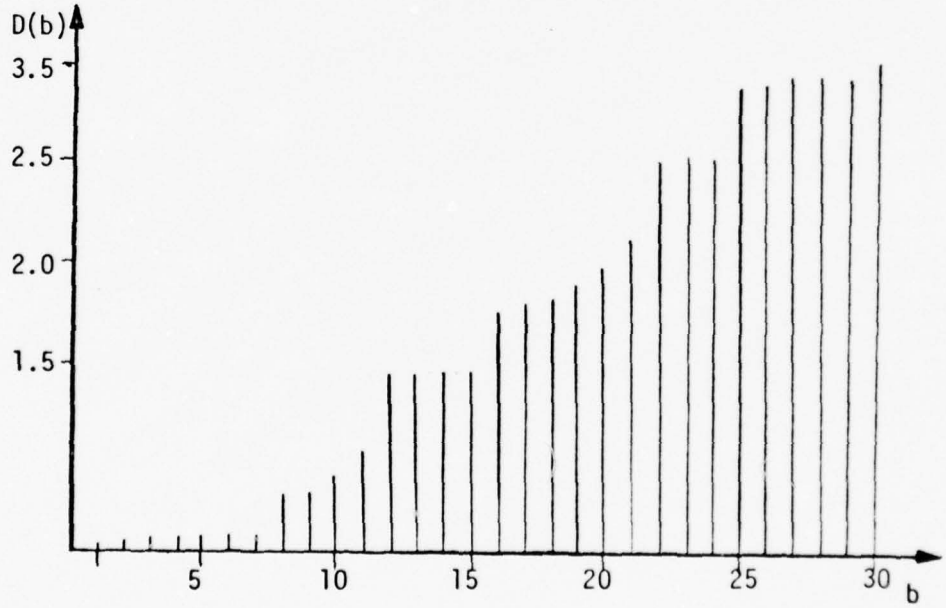


(d)

Figure 4-3 (continued)

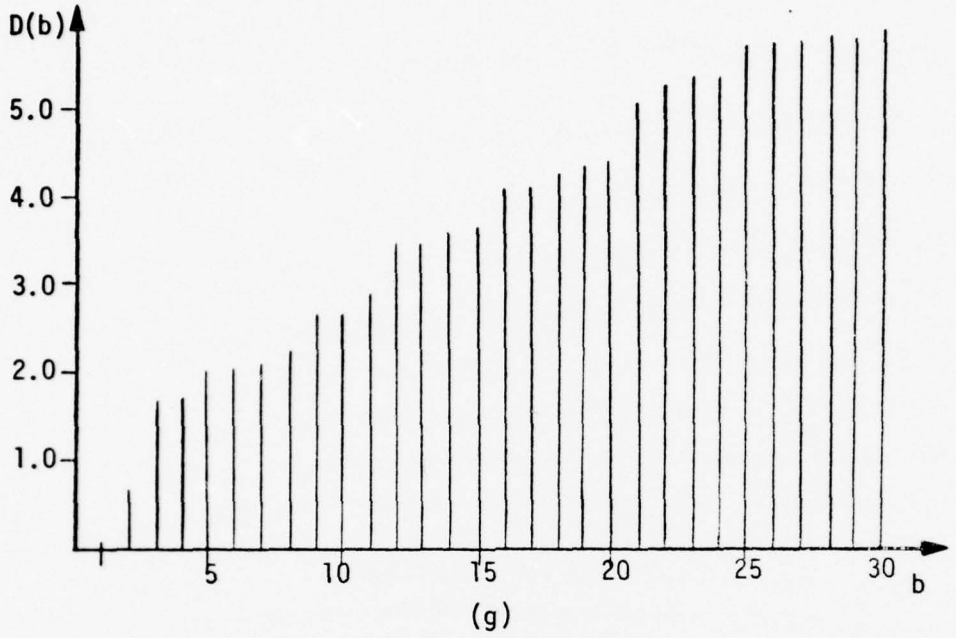


(e)

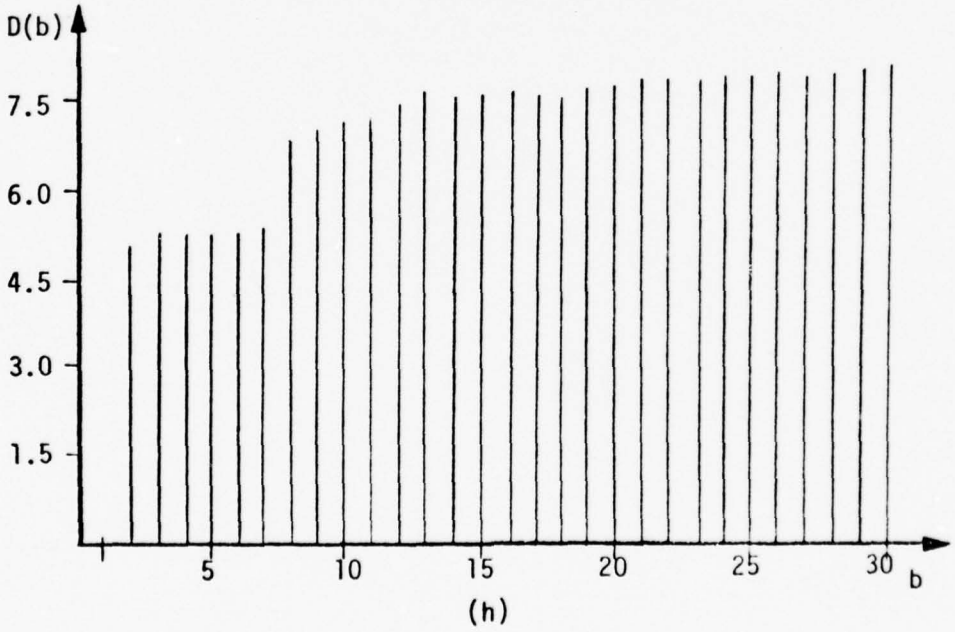


(f)

Figure 4-3 (continued)



(g)



(h)

Figure 4-3 (continued)

previously chosen which represented at least three classes. This procedure resulted in the eight moments pairs shown in Table 4-3.

According to the Mahalanobis distance criterion, these eight descriptors will produce better classification results than any other set of eight chosen from the thirty listed in Table 4-2. It should be noted that if all possible combinations of eight dimensions of the original thirty were tested separately by experiment to determine the best eight, a total of over 5.8 million experiments would have to be performed.

In evaluating the classification results obtained with these eight moment pairs, the fact that all of the aircraft were considered simultaneously should be remembered. If the Mahalanobis distance criterion were applied to a subset of the classes, a different set of descriptors might very well be chosen. Indeed in Experiment No. 5, only four moment pairs were used when considering two of the aircraft listed in Table 4-1, p. 59.

A representative sample of the experiments conducted using the foregoing radar data and recognition system follows:

Experiment No. 1

In this experiment, two classes were formed by dividing the aircraft in Table 4-1 according to their classification as fighter or transport. The moment pairs listed in Table 4-3 were calculated for each pattern. The resulting transport class contained 50 patterns, while the fighter class contained 53 patterns.

The separability of the classes could not be determined by the LMSE algorithm, since the present implementation of this algorithm

TABLE 4-3. The Most Discriminant Moment Pairs As Determined From Table 4-2 By the Mahalanobis Distance Criterion

Order	X Index	Y Index
2	0	2
2	2	0
4	4	0
4	0	4
5	3	2
6	3	3
6	0	6
6	5	1

only allows for a maximum of 95 patterns in both classes. This maximum was imposed by memory limitations.

Training was first carried out using the perceptron algorithm. After 100 iterations, the best result classified 102 patterns correctly out of the 103 total. After another 26 iterations, the algorithm converged, yielding 100% proper classification results when using the original training set.

A new set of data was constructed by randomly selecting twenty patterns from the original training set, and either deleting or adding a single reflection point. Ten patterns were created by adding a reflection point, the other half by deleting a reflection point. These alterations to the original set were designed to simulate the addition or deletion of specific pieces of equipment; such as spare fuel tanks, or externally-mounted weapons.

The new data set was presented to the weight vectors as determined by the perceptron algorithm for the original set of training samples. The resulting decision functions misclassified two of the twenty patterns, resulting in a 10% rate of error for new data. The twenty new patterns were then added to the training set, and the perceptron algorithm converged again after an additional 300 iterations.

The same tests were conducted using the Bayes classifier. The classifier was trained using equal a priori probabilities of p(fighter)=p(transport)=1/2. The training set of patterns was then presented to the classifier, with only one of the 103 patterns being misclassified. This is a classification error of approximately .97%.

The new data set which was created for use with the perceptron was presented to the classifier. For this part of the experiment, the Bayes classifier properly assigned 17 of the new patterns to their respective classes, for a classification error of 15%. The Bayes classifier was then retrained by including the new data in the training set. All 103 patterns were then checked for classification, with only one error occurring. This represents an error of only .81% after retraining.

Experiment No. 2

For this experiment, the aircraft categorized as transports were divided into two classes. The patterns which represent the C5A and the An-22 aircraft formed one class, while the patterns of the TU-22 and B-737 made up the second class. This division was based on the large difference in wingspans exhibited by the two types of aircraft. The two classes represented a total of 50 patterns, 25 in each class.

When the two classes were presented to the LMSE algorithm, the error vector proved the class to be not linearly separable. The classes were then input to the Bayes algorithm for training. By assigning equal a priori probabilities of one half to both classes, the classification error incurred was 2% due to one pattern being misclassified. Since the Bayes classifier produces a second degree (hyperquadric) decision boundary, these classification results proved much better than those produced by the linear decision functions of either the LMSE or perceptron algorithms. These pattern classes were then presented to the perceptron algorithm for training. After

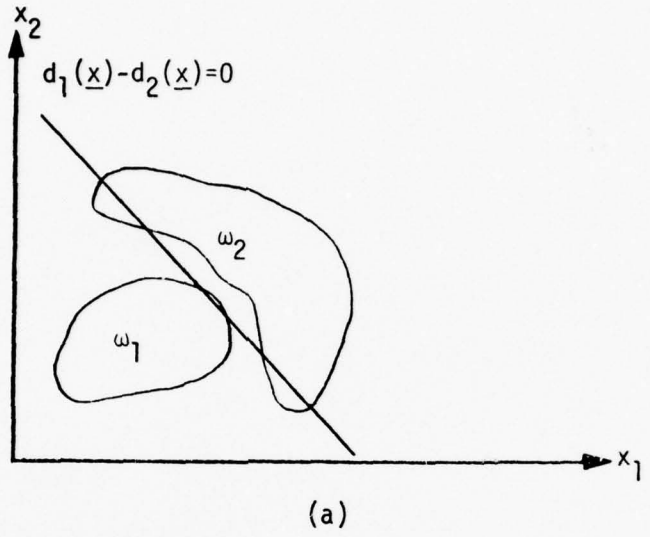
approximately 5000 iterations, the best result was 43 out of 50 patterns classified correctly. Since the classes were not linearly separable, no new data were created to test the decision functions obtained by the perceptron algorithm.

While the Bayes classifier yielded acceptable results in this experiment, it was hoped that a linear decision boundary could be used to dichotomize the classes properly. This motivated the division of one of the classes into subclasses as discussed in the next experiment.

Experiment No. 3

This experiment was based on the same patterns as Experiment No. 2, with the exception that the transport class with the larger wingspan was divided into two subclasses. This produced three classes called "small transports," C5A, and AN-22. The classification results were considered correct if a C5A or an AN-22 pattern was assigned either the C5A or AN-22 classes, and incorrect only if assigned to the class of small transports. This effectively reduced the use of the classifier to that of a two class case, but allows three decision boundaries in the feature space.

Figure 4-4 shows a hypothetical problem where this approach is useful. In Figure 4-4(a), the two classes are not linearly separable as shown by the decision boundary $d(\underline{x}) = d_1(\underline{x})-d_2(\underline{x})=0$. In Figure 4-4(b), it is seen that a piecewise linear decision boundary will properly dichotomize the classes. This form of a decision boundary can be obtained by dividing the patterns in ω_2 into two classes, say ω_{2a} and ω_{2b}. The two decision boundaries formed by $d(\underline{x}) = d_1(\underline{x})-d_{2a}(\underline{x})=0$



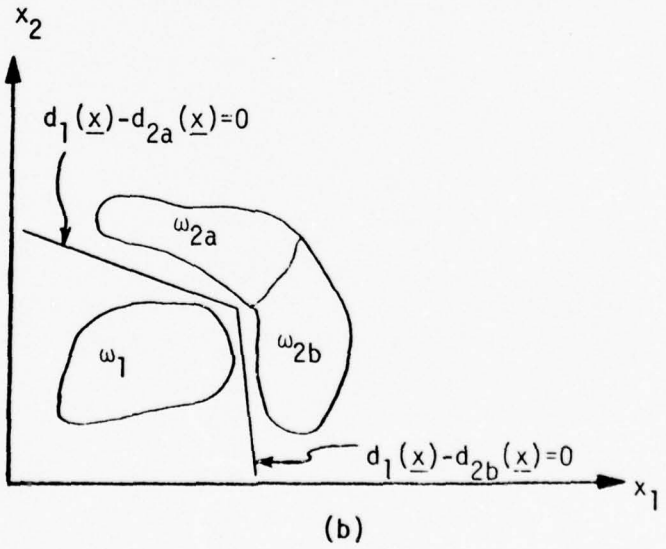


Figure 4-4. Two classes which are not linearly separable (a), become separable when ω_2 is divided into two subclasses, ω_{2a} and ω_{2b} in (b).

and $d(\underline{x}) = d_1(\underline{x})-d_{2b}(\underline{x})$ now separate the two classes.

Since the LMSE algorithm is only suitable for use with two classes at a time, the classes were presented pairwise to the algorithm. The classes proved to be pairwise separable, meaning that the classes could be properly dichotomized using three decision boundaries.

A solution was achieved by the perceptron algorithm after 7000 iterations of the training sequence. A new data set was created by adding and deleting reflection centers as discussed in Experiment No. 1. The resulting 20 patterns were classified using the decision functions obtained from the original training data. Three patterns of 20 were misclassified while using these decision functions.

The perceptron algorithm was then retrained, with the 20 new patterns added to the original data set. After an additional 2000 iterations, 69 of the 70 patterns were properly classified. This is only a 1.43% classification error, once the algorithm had seen the new data.

The Bayes classifier was then trained on the original 50 patterns. A 25% a priori probability was assigned to the C5A and AN-22 classes, while 50% was assigned to the small transport class. Classification tests were then conducted using the original data. Only one out of 50 patterns was misclassified, yielding an error of 2% for the chosen probabilities.

The altered patterns were then used in testing the Bayes decision function. Only one pattern was misclassified out of the 20. After retraining with the altered aircraft added to the training set, the pattern which produced the error in the original data set was the only airplane misclassified.

Experiment No. 4

The objective of this experiment was to obtain a set of decision functions capable of discriminating between the transport aircraft listed in Table 4-1, page 59. A total of 50 patterns in the four classes comprised the training set.

The perceptron algorithm trained on the four pattern classes for over 7000 iterations. At the end of the training period, 56% of the patterns were properly classified.

The training patterns were then input to the Bayes classifier. Equal a priori probabilities were assigned to each of the four pattern classes. When the training patterns were classified with these decision functions, four of the 50 patterns were misclassified. The decrease in error which the Bayes exhibits over the perceptron is due to the hyperquadric decision boundary of the Bayes classifier.

Since the pattern classes could not be proved linearly separable using the decision functions generated by the perceptron, it was necessary to obtain decision functions by considering the classes pairwise. This was accomplished by presenting the classes to the LMSE algorithm two at a time. The algorithm converged to a solution in each of the six possible combinations. This means that 100% proper classification can be achieved by using the six decision functions produced by this algorithm.

An alternate method of subdividing the data to achieve the desired classification results is to first present the unknown patterns to the decision functions obtained in Experiment No. 3, classifying the patterns as either large or small transports. Then based upon the results of that

classification, each pattern could be input to the decision functions derived by the LMSE algorithm to make the final classification. This method is illustrated as a flow chart in Figure 4-5.

The results of the four preceding experiments are presented in Table 4-4.

Experiment No. 5

The objective of this experiment was to compare the classification results of the automatic recognition system with the results obtained by using human interpreters. For this example, only two aircraft were used, the FB-111A and the B-737. These two aircraft were chosen since the FB-111A has the largest wingspan of any fighter and the B-737 has the smallest wingspan of any transport in Table 4-1, page 59. The moment pairs used as descriptors were {0,2}, {2,0}, {1,3}, and {4,1}.

Fourteen patterns from each class comprised the training set. Photographs were taken of each training pattern and were labeled according to the class to which they belonged. The interpreters were given the 28 photographs approximately 5 minutes before the experiment began, allowing them time to examine each one carefully. Examples of the photographs are shown in Figure 4-6. Figure 4-7 contains reduced photographs of the patterns which represent the FB-111A training patterns, while Figure 4-8 contains those for the B-737 class.

Both the perceptron and the Bayes classifiers were trained using the same data as that given the interpreters. Equal a priori probabilities were assigned to each class for the Bayes classifier. The perceptron algorithm converged to a solution in six iterations.

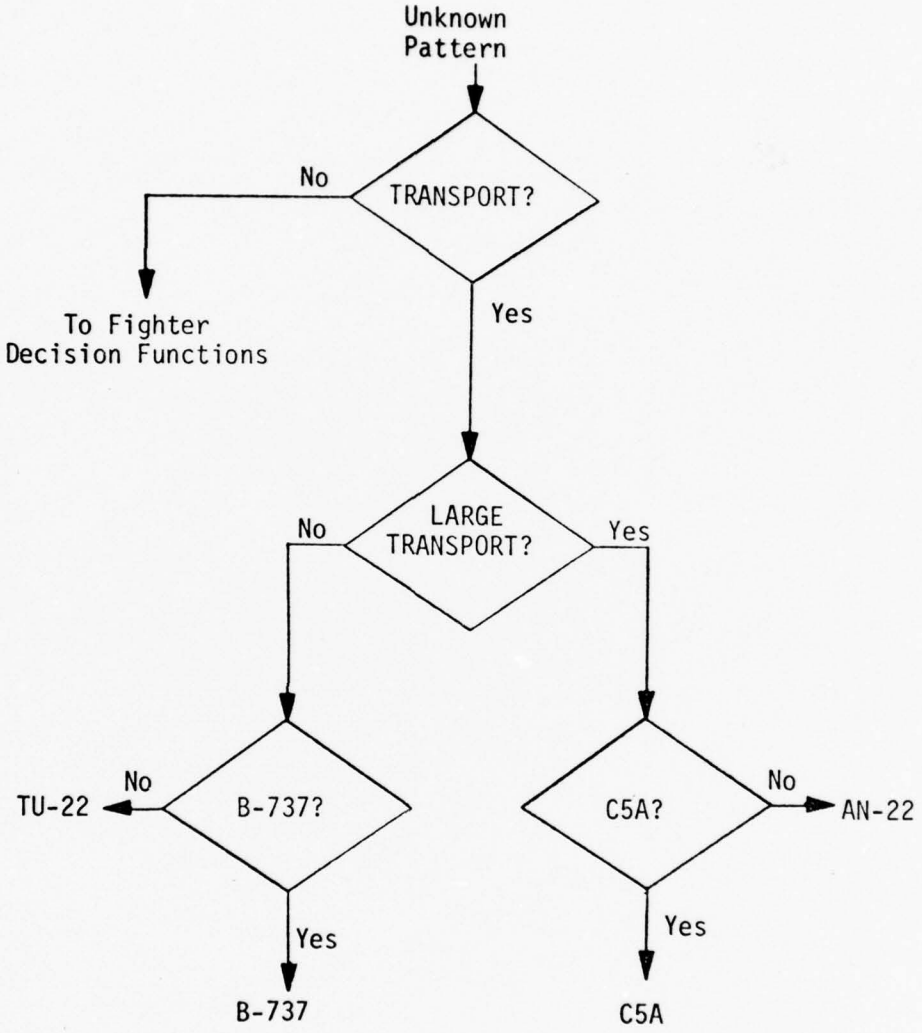


Figure 4-5. Flow graph representation of the classification scheme as outlined in Experiment No. 4.

TABLE 4-4. Compiled Classification Results

Experiment/ Algorithm	Percent Correct Classification		
	Training Data	Altered Data	Altered Data After Training
Experiment No. 1			
Bayes	99.13	85.00	99.19
LMSE	N/A	N/A	N/A
Perceptron	100.00	90.00	100.00
Experiment No. 2			
Bayes	98.00	N/A	N/A
LMSE	N/S	N/A	N/A
Perceptron	86.00	N/A	N/A
Experiment No. 3			
Bayes	98.00	95.00	98.57
LMSE	N/A	N/A	N/A
Perceptron	100.00	85.00	98.57
Experiment No. 4			
Bayes	92.00	N/A	N/A
LMSE	N/A	N/A	N/A
Perceptron	56.00*	N/A	N/A
Perceptron/LMSE	100.00**	N/A	N/A

N/S = Not separable; N/A = Not available.

*Resulting considering all classes simultaneously.

**Results considering classes pairwise.

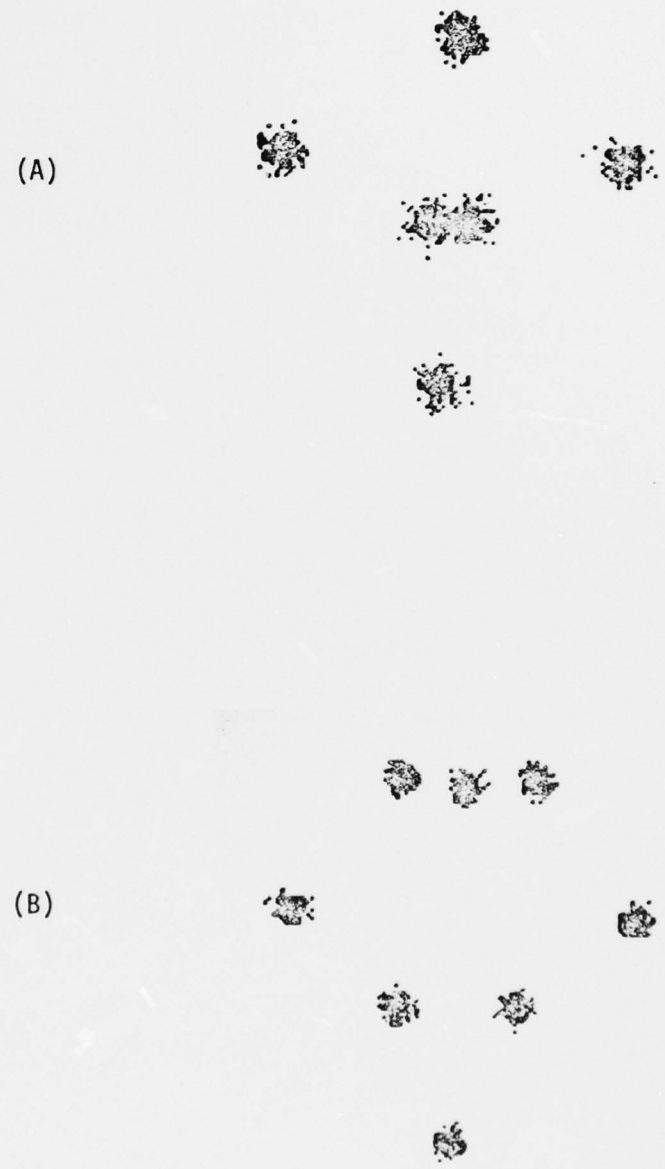


Figure 4-6. Samples of the photographs given the human interpreters. (A) Representative of FB-111A class; (B) represents the B-737 class.

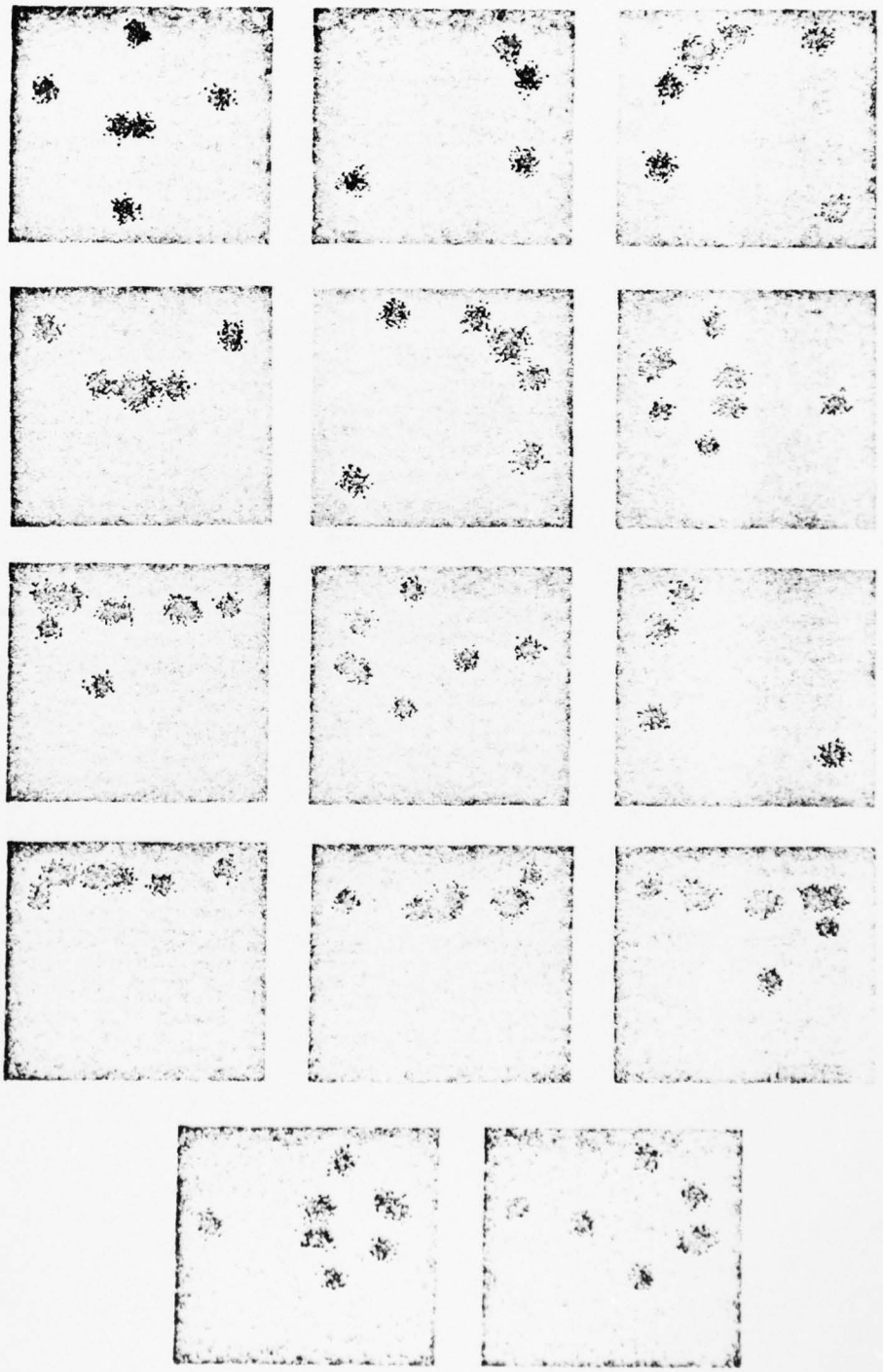


Figure 4-7. These are reduced photographs of all the training patterns for the FB-111A class.

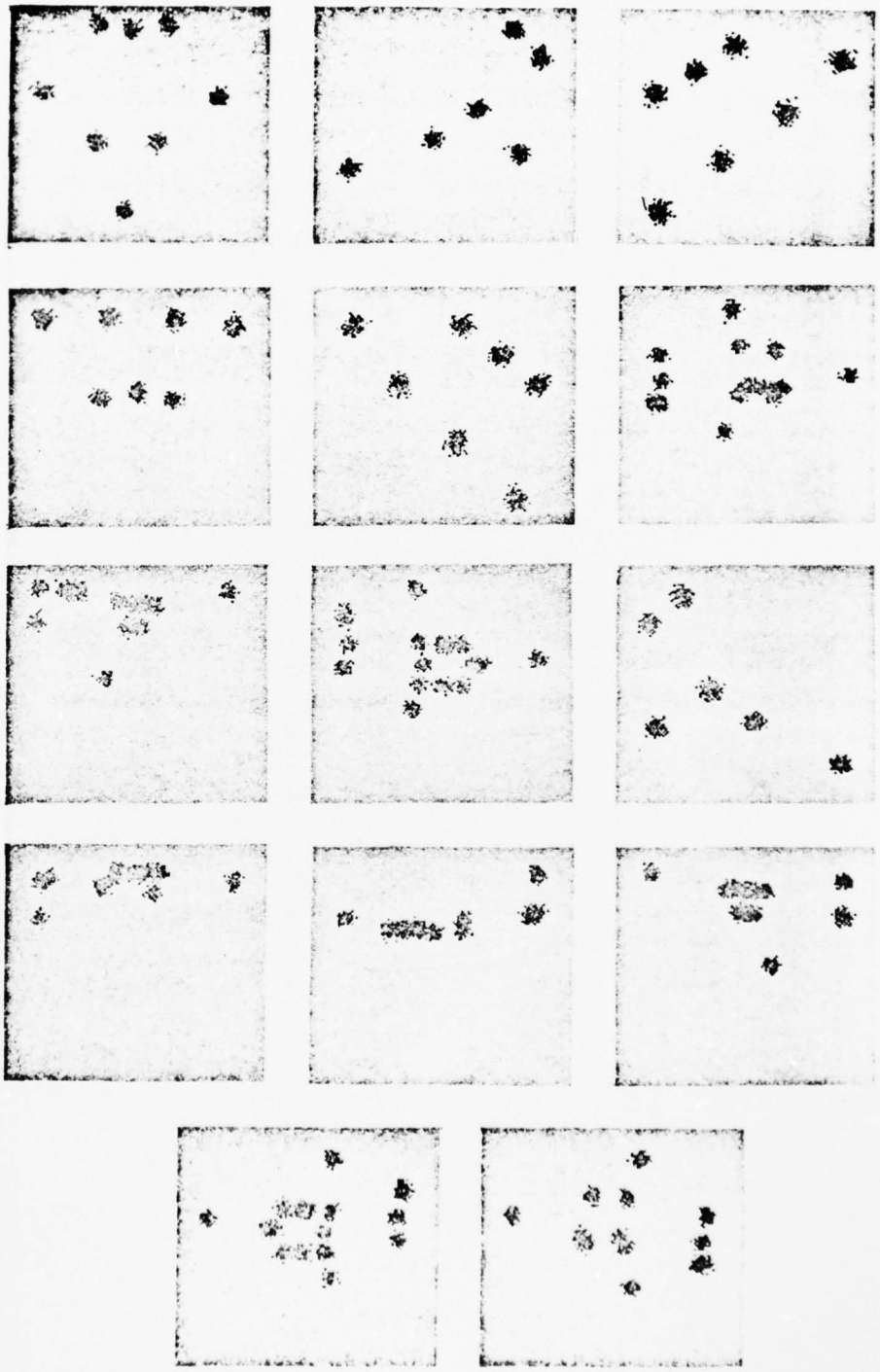


Figure 4-8. Reduced photographs of the training patterns for the B-737 class.

Two subjects were used in the experiment, one of which had been actively involved in the project since its beginning; the other had little or no experience with radar images or pattern recognition. The training patterns were presented to the interpreters, allowing them as much time as needed to make their decision. The experienced interpreter gave 100% proper classifications on the training set, while the inexperienced interpreter had an 18% error rate.

An altered data set was then created and presented to both interpreters. The photographs were left available to the interpreters for reference. The experienced interpreter had a 10% rate of error, and the inexperienced interpreter misclassified 20% of the new patterns. These results are based on twenty observations.

When the patterns were presented to the recognition system, 100% proper classification was recorded for both classifiers on the training set. The altered data produced a 10% error rate for the Bayes classifier, but the perceptron decision function separated the classes with 100% accuracy.

The results of this experiment are summarized in Table 4-5.

Experiment No. 6

In Chapter 3, a first order approximation of the Mahalanobis distance was presented which was based on the assumption that the data being used were decoupled. In order to gain further insight into the error involved in this approximation, two examples are presented below.

In the first example, the Mahalanobis distance and its approximation is calculated for the patterns of two classes. The patterns used

TABLE 4-5. Classification Error Rates for Experiment No. 5

Classifier	Training Pattern	Altered Patterns
Perceptron	0%	0%
Bayes	0%	10%
Experienced Interpreter	0%	10%
Inexperienced Interpreter	18%	20%

are eight dimensional vectors. Figure 4-9(a) shows the true Mahalanobis distance calculated for an eight dimensional data set. The approximate distance is illustrated in Figure 4-9(b). The values of the Δ_b's given in Equation (3-16) and the error involved is listed in Table 4-6. While the amount of error incurred by the approximation is relatively large, it should be noted that the general shapes of the graphs in Figure 4-9(a) and (b) are very similar.

As a second example of the estimation error, Figure 4-10(a) shows a graph of the true Mahalanobis distance between the means of two classes, while Figure 4-10(b) is a graph of the approximation using the decoupled assumption. The estimated distances are again different from their exact counterparts, but almost every large Δ_b in the true Mahalanobis distance is characterized by a large $\hat{\Delta}_b$ in the estimate. It is of interest to note that while it required over an hour of computer time to calculate the values of the true Mahalanobis distance, calculation of the estimate was completed in less than five minutes.

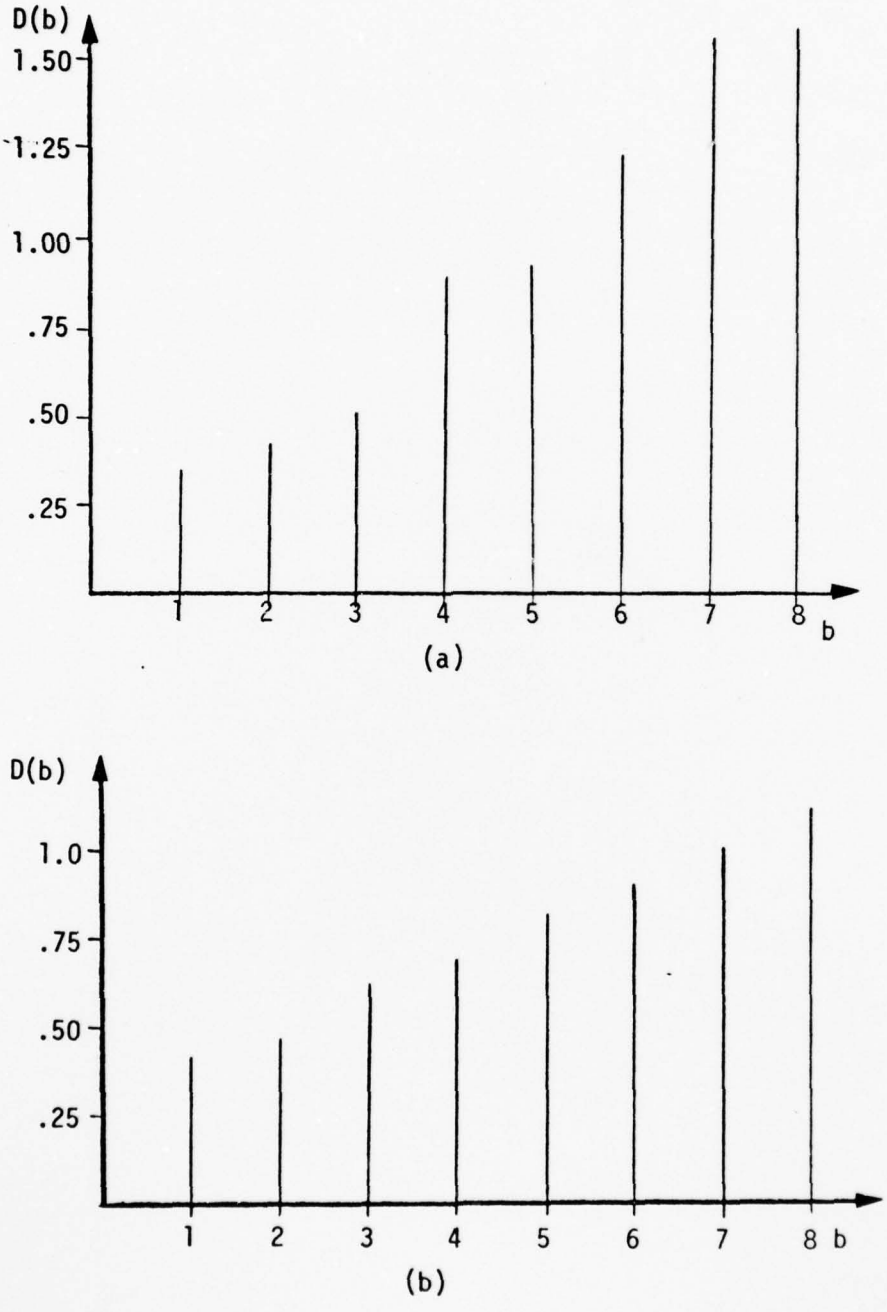


Figure 4-9. The true Mahalanobis distance is shown in (a), while (b) is a plot of the estimated distance as a function of increasing dimension.

TABLE 4-6. Comparison of the Actual and Estimated Mahalanobis Distances

b	Δ_b	$\hat{\Delta}_b$	Error
1	.4120	.2436	40.9%
2	.0593	.0187	68.5%
3	.1511	.0523	65.4%
4	.079	.2579	22.6%
5	.0011	.0108	88.2%
6	.1464	.1907	30.3%
7	.1511	.1979	30.9%
8	.1029	.0371	63.9%

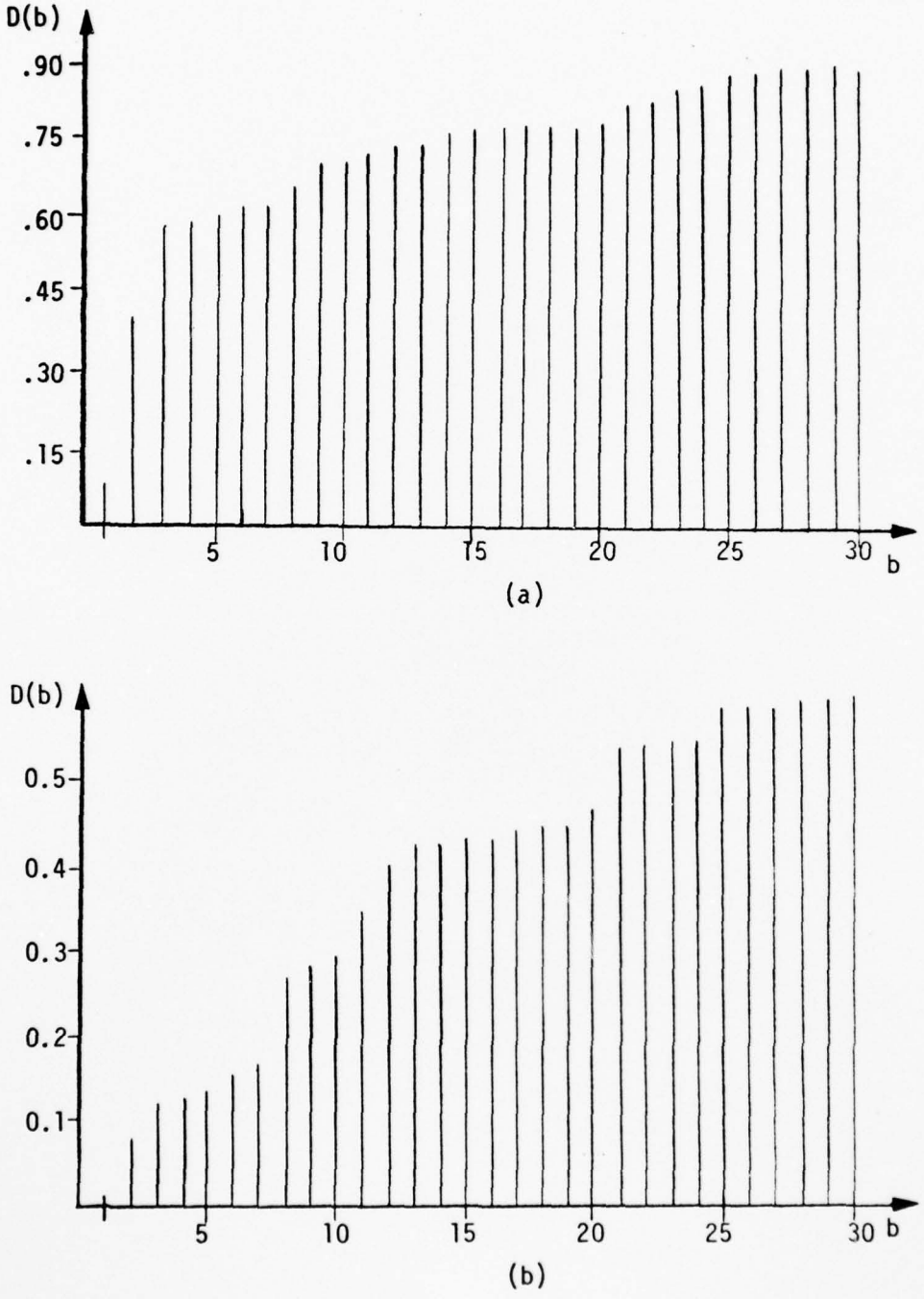


Figure 4-10. The exact Mahalanobis distance is shown in (a), while (b) represents the approximate distance.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

The development of a recognition system has been presented along with a method for the selection of pattern features. The system has been applied to radar images, thus testing the descriptive abilities of two dimensional moment pairs on unstructured data.

The experiments presented in Chapter 4 indicate the usefulness of moments as descriptors when coupled with decision-theoretic classification techniques. Experiment No. 1 is illustrative of the ability of the system to classify the radar images of fighters against bombers and cargo planes. This experiment was successful in showing the ability of the system to adapt to variations in the aircraft used for training. The results of this experiment along with the other tests show reasons for being optimistic about the capabilities of the system when real data become available for experimentation.

In comparing the classification results of the system to those of human interpreters, the system produced equivalent, or better, results depending upon the experience of the interpreter. The interpreters required anywhere from one to three minutes in arriving at their decisions, while the computer required a few seconds for classification, including the time necessary for preprocessing. The time consumed in producing the decision alone was under a second. The usefulness of such a system for radar surveillance installations at remote sites where environmental conditions might prohibit human habitation is evident. It should also be noted that during the evaluation experiment cited in

Chapter 4, the interpreters were given photographs of the training patterns. As the number of training patterns is increased significantly, it becomes impractical to provide photographs of all the patterns to the humans, while a larger number of images can easily be stored in a computer. Under these conditions, it is felt that the autonomous recognition system would perform as well or better than an interpreter, both in terms of accuracy and classification speed.

During the development of the system, several difficulties were encountered. In the experiments with the Bayes classifier, the covariance matrix was often ill conditioned. In the present implementation, this was handled with a double-precision subroutine which utilizes a double pivoting strategy, but it can be shown (Forsythe and Moler, 1967) that for a positive definite matrix the pivoting strategy $\underline{p}^T = (1,2,3, \ldots,n)$ will produce an acceptably small error. By developing an inverse using this strategy, a significant savings in time and memory could be realized. It should also be noted that certain assumptions were made in the development of the Bayes classifier which may not be met when implemented. This means that at times, the algorithm may not produce an acceptable solution even if one exists. An example of this is given in Experiment No.'s 1 through 4. On the average over the four experiments, the Bayes classifier produced very good results. In Experiments 1 and 3, however, the classes were known to be separable but the Bayes produced classification errors on the training data. Even when the a priori probabilities were biased to produce proper classification, the error still occurred.

While the perceptron algorithm produced good classification results

during experimentation, the training times were generally much longer than with the Bayes classifier. This was due mainly to the algorithm being implemented on a minicomputer which required that the training patterns be swapped in and out of memory from disk. By running the algorithm on a large computer, we believe that the execution times could be reduced at least by an order of magnitude. Training limitations, however, should not be confused with classification performance. Once the training phase has been completed, both the Bayes and perceptron classifiers can be implemented to achieve speeds approaching real-time operation.

The first order approximation to the Mahalanobis distance should be refined to give a more accurate representation of the true distance. Two possible solutions to this problem are as follows. If an updating method could be developed for estimating the inverse of the covariance matrix, then the Mahalanobis distance could be estimated to the accuracy of the inverse matrix. While this has not been studied in depth, the properties of a positive definite, symmetric matrix such as the covariance matrix makes this an attractive approach to the estimation problem. A second approach might be to express the Mahalanobis distance as a power series, where the accuracy of the estimate will only be limited to the number of terms taken in the series.

While the experiments discussed in Chapter 4 give good reason to be optimistic about the approach taken thus far as it relates to radar images, some additional experiments should be conducted. First, the system should be tested using real data. Although great care

was taken to model images as closely as possible to real data, modeling is certainly no substitute for real radar data.

The performance of the system needs to be compared further with the capabilities of trained interpreters. Such a comparison could prove very valuable in evaluating the system's performance prior to field installation.

REFERENCES

REFERENCES

Alt, F. L. [1962]. "Digital Pattern Recognition by Moments," Optical Character Recognition (G. L. Fisher, D. K. Pollock, B. Raddack, and M. E. Stevens, eds.), Spartan Books, New York.

Anderson, T. W. [1958]. Introduction to Multivariate Statistical Analysis, John Wiley and Sons, New York.

Casey, R. G. [1970]. "Moment Normalization of Handprinted Characters," IBM Journal Research Development, vol. 14, no. 5, pp. 548-557.

Duda, R. O. [1970]. "Elements of Pattern Recognition," Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications (J. M. Mendel and K. S. Fu, eds.), Academic Press, New York.

Cooley, W. W., and Lohnes, P. R. [1971]. Multivariate Data Analysis, John Wiley and Sons, New York.

Forsythe, G., and Moler, C. B. [1967]. Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Giuliano, V. E., Jones, P. E., Kimball, G. E., Meyer, R. F., and Stein, B. A. [1961]. "Automatic Pattern Recognition by a Gestalt Method," Information and Control, vol. 4, no. 4, pp. 332-345.

Gonzalez, R. C., and Wintz, P. W. [In Press]. Digital Image Processing Fundamentals, Addison-Wesley Publishing Co., Reading, Massachusetts.

Hue, M. K. [1962]. "Visual Pattern Recognition Using Moment Invariants," IRE Transactions on Information Theory, vol. IT-8, pp. 179-187.

Ledley, R. S. et al. [1965]. "FIDAC: Film Input to Digital Automatic Computer and Associated Syntax-Directed Pattern Recognition Programming System," in Optical and Electro-Optic Information Processing Systems (J. Tippet, D. Beckowitz, L. Clapp, C. Koester, and A. Vanderburgh, Jr., eds.), MIT Press, Cambridge, Massachusetts, Chapter 33.

Noble, Ben [1969]. Applied Linear Algebra, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Peebles, P. Z. [1976]. In private conversation with author.

Rosenblatt, F. [1957]. "The Perceptron: A Perceiving and Recognizing Automaton," Project PARA, Cornell Aeronaut. Lab. Rept. 85-460-1.

Smith, F. W., and Wright, W. M. [1971]. "Automatic Shop Photointerpretation by the Method of Moments," IEEE Transactions on Computers, vol. C-20, no. 9, pp. 1089-1094.

Tou, J. T., and Gonzalez, R. C. [1974]. Pattern Recognition Principles, Addison-Wesley Publishing Co., Reading, Massachusetts.

Young, T. Y., and Calvert, T. W. [1974]. Classification, Estimation, and Pattern Recognition, American Elsevier Publishing Co., Inc., New York.

APPENDICES

APPENDIX A

THE HOTELLING TRANSFORMATION

By extracting the coordinate points of an image and creating a set of vectors of the form

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{A-1}$$

it is desired to obtain a corresponding set of vectors

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{A-2}$$

by means of the linear transformation

$$\underline{y} = \underline{A}\underline{x}. \tag{A-3}$$

The rows of the matrix $\underline{A}$ are the eigenvectors of the covariance matrix

$$\underline{C}_x = E\{(\underline{x} - \underline{m}_x)(\underline{x} - \underline{m}_x)^T\}, \tag{A-4}$$

where $\underline{m}_x$ is the mean vector of the image.

One of the basic concepts of the transformation is that the eigenvectors point in the direction of maximum variance of the data. This concept leads to the use of this transformation for standardizing the slant of an image, since most objects have a maximum variance in a single direction along the object. Figure A-1 shows the original coordinate system x and the eigenvectors labeled y. If the eigenvectors are normalized, then to rotate the image by the angle θ, the transformation matrix is given by

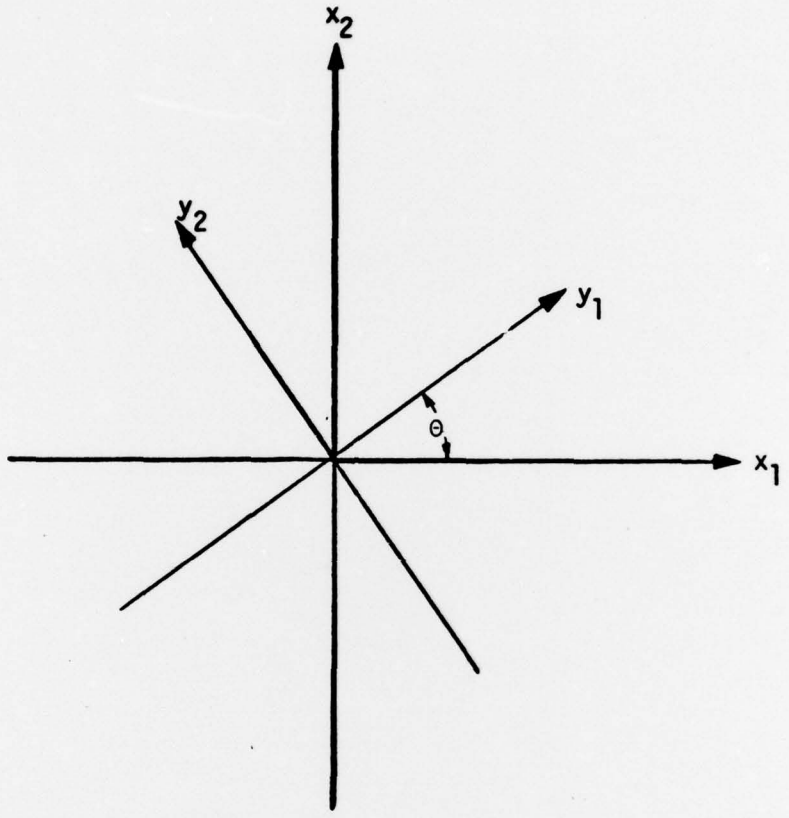


Figure A-1. Rotation of coordinate system.

$$\underline{A} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} \qquad \text{(A-5)}$$

where e_{ij} is the jth component of the ith eigenvector. If it is desired to standardize the image against translation, Equation (A-3) may be rewritten as

$$\underline{y} = \underline{A}(\underline{x} - \underline{m}_x). \qquad \text{(A-6)}$$

It is important to note that, if $\underline{e}_1$ and $\underline{e}_2$ are valid eigenvectors of $\underline{C}_x$, then the pairs $(\underline{e}_1, -\underline{e}_2)$, $(-\underline{e}_1, \underline{e}_2)$, and $(-\underline{e}_1, -\underline{e}_2)$ are also valid eigenvectors, but only two pairs represent right-handed coordinate systems. Figure (A-2) shows the effect of allowing a pair of eigenvectors which represent a left-handed coordinate system to be used for the transformation. Of course, if the eigenvectors are checked to see if they form a right-handed system, then only two possible orientations remain for the transformed image regardless of the original orientation.

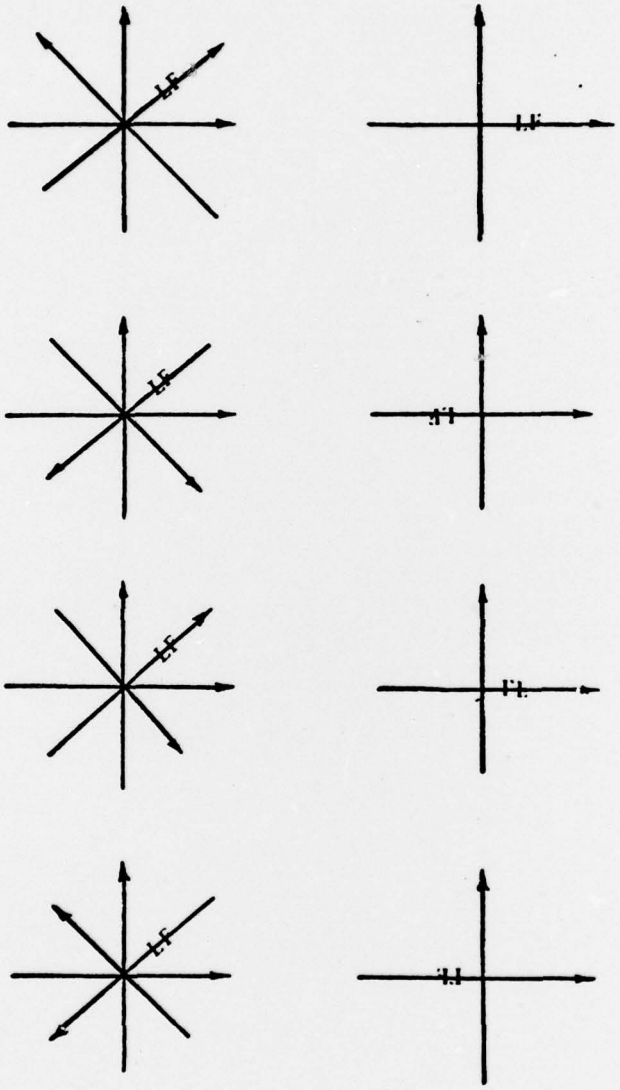


Figure A-2. The direction of valid eigenvectors are given in (a), while (b) shows the effect of the rotation. Only the first two sets of eigenvectors form a right-handed coordinate system.

APPENDIX B

MAHALANOBIS DISTANCE OF DECORRELATED DISTANCE

It is desired to show the Mahalanobis distance of a data set which has undergone a linear transformation to decorrelate the data is equal to the distance measure of the original data. The Mahalanobis distance of the original data is given by

$$D_x = (\underline{x}-\underline{m}_x)^T \underline{C}_x^{-1}(\underline{x}-\underline{m}_x), \tag{B-1}$$

while the distance measure for the decorrelated data is given by

$$D_y = (\underline{y}-\underline{m}_y)^T \underline{C}_y^{-1}(\underline{y}-\underline{m}_y). \tag{B-2}$$

The covariance matrix for the transformed data can be written as

$$\begin{aligned}\underline{C}_y &= E\{(\underline{y}-\underline{m}_y)(\underline{y}-\underline{m}_y)^T\} \\ &= E\{(\underline{Ax}-\underline{Am}_x)(\underline{Ax}-\underline{Am}_x)^T\} \\ &= E\{\underline{A}(\underline{x}-\underline{m}_x)(\underline{x}-\underline{m}_x)^T \underline{A}^T\} \\ &= \underline{AC}_x\underline{A}^T. \end{aligned} \tag{B-3}$$

The Mahalanobis distance in the transformed space can be expressed as

$$\begin{aligned}D_y &= (\underline{y}-\underline{m}_y)^T \underline{C}_y^{-1}(\underline{y}-\underline{m}_y) \\ &= (\underline{Ax}-\underline{Am}_x)^T (\underline{AC}_x\underline{A}^T)^{-1} (\underline{Ax}-\underline{Am}_x) \\ &= (\underline{x}-\underline{m}_x)^T \underline{A}^T (\underline{A}^T)^{-1} \underline{C}_x^{-1} \underline{A}^{-1} A(\underline{x}-\underline{m}_x) \\ &= (\underline{x}-\underline{m}_x)^T \underline{C}_x^{-1}(\underline{x}-\underline{m}_x) \\ D_y &= D_x, \end{aligned} \tag{B-4}$$

which is the desired result.